Apama on Cloud Foundry

Version 10.1

December 2017

Document ID: CFC-PAM-UG-101-20171228

Table of Contents

About This Guide....5
- Document Conventions.... 5
- Online Information.... 6

Architecture and Components of Apama Deployment for Cloud Foundry....7

Getting Started with Apama for Cloud Foundry....9

Collecting Pre-installation Data....11
- Network Configuration.... 12
- BOSH Virtual Machine Configuration.... 12
  - Service Broker BVM Configuration.... 13
  - Service Instance BVMs Configuration....14
  - Compilation BVMs Configuration....15
- User Roles.... 15
- Cloud Foundry Security Groups....16
- Firewall Settings Requirements....16
  - Firewall Settings for BVMs....16
  - Firewall Settings for User Roles....21

Installing Apama for Cloud Foundry.... 23

Configuring the Apama Security Manager....25

Defining Service Plans....29
- Default Service Plans....30
- Using the Service Plan Script.... 30

Deploying Apama to Cloud Foundry using BOSH....35
- Before You Begin....40
- Customizing Your Deployment.... 40
  - Modifying the setup.properties File.... 40
  - Customizing the YAML Files.... 41
    - cloud-config.yml....42
    - cf-Apama.yml....44
    - custom-vars.yml....47
  - Creating Additional Customizations....48
  - Customizing the creds.properties File....49
- Deploying Apama Using the boshDeploy Script.... 50

Creating and Adding the Apama Buildpack to Cloud Foundry.... 51

Installing Apama Fixes....53

Upgrading Apama on Cloud Foundry.......... 55

Using Apama on Cloud Foundry.......... 57
Creating an Apama Service.......... 58
Creating a Service Key.......... 59
Client Access to the Apama Service.......... 60
Injecting the Apama Project.......... 61
Injecting the Apama Project by Using the Apama Buildpack.......... 61
Injecting the Apama Project by Using the CLI Tool.......... 63
Collecting Diagnostic Information.......... 65
Extending Logging.......... 66
Collection of Configuration Files.......... 66
Collection of Log Files.......... 67

About This Guide

This guide explains how to configure and deployApama on Cloud Foundry using BOSH, and covers the capabilities provided for Apama Cloud Foundry enablement.

Note that this guide supplements the Apama product documentation. You must have preliminary knowledge of Apama and Cloud Foundry BOSH to perform the procedures described.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies storage locations for services on webMethods Integration Server, using the convention *folder.subfolder:service* .
UPPERCASE	Identifies keyboard keys. Keys you must press simultaneously are joined with a plus sign (+).
Italic	Identifies variables for which you must supply values specific to your own situation or environment. Identifies new terms the first time they occur in the text.
`Monospace font`	Identifies text you must type or messages displayed by the system.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at http://documentation.softwareag.com. The site requires Empower credentials. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

You can find product information on the Software AG Empower Product Support website at https://empower.softwareag.com.

To submit feature/enhancement requests, get information about product availability, and download products, go to Products.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the Knowledge Center.

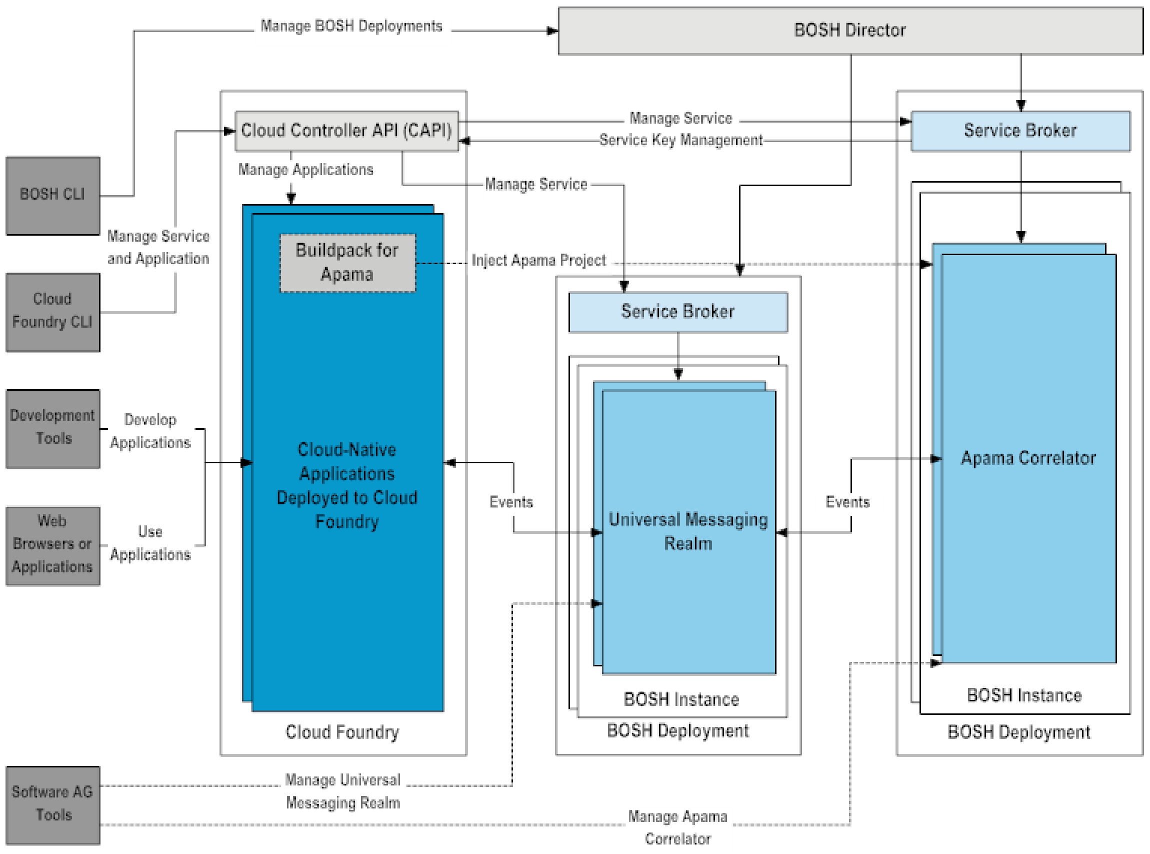
Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at http://techcommunity.softwareag.com. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

1 Architecture and Components of Apama Deployment for Cloud Foundry

You can deploy Apama by creating a Cloud Foundry BOSH deployment on a set of BOSH virtual machines (BVMs). The following diagram shows the architecture of an Apama with Universal Messaging deployment for Cloud Foundry and the communication paths involved in the deployment.



When you deploy Apama to Cloud Foundry, you can use it as a cloud native-application and access it through a web browser, mobile and desktop applications, or other cloud-native applications. The following sections contain instructions on how to configure the network for deployment, the firewall, and the BVMs for different environments.

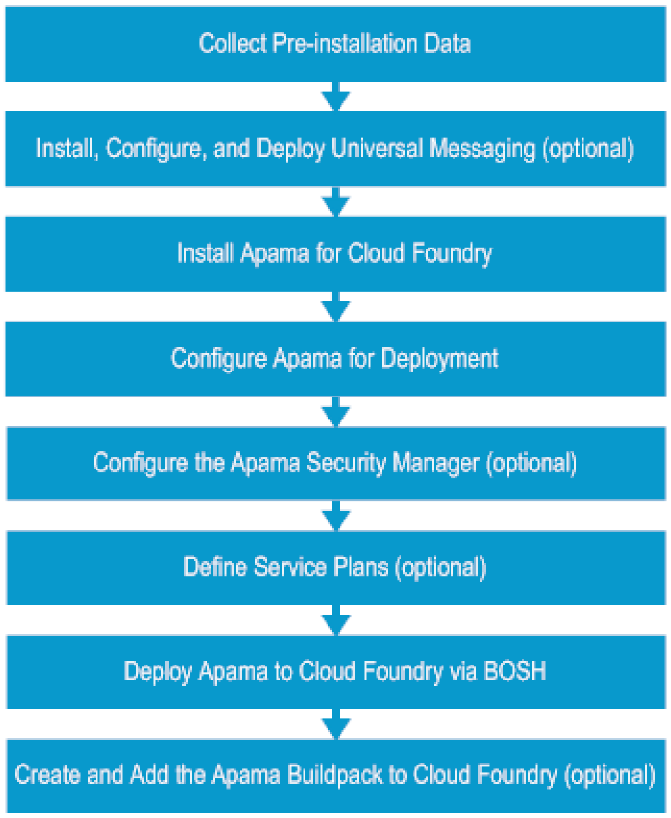
For information about the following components, see:

Component	Reference
BOSH Director	BOSH Cloud Foundry documentation at https://bosh.cloudfoundry.org/docs

Component	Reference
Cloud Controller API, Service Broker	Cloud Foundry documentation at https://docs.cloudfoundry.org/
Apama Correlator	*Introduction to Apama*
Universal Messaging Realm	*Universal Messaging Administration Guide*

2 Getting Started with Apama for Cloud Foundry

The process of deploying Apama for Cloud Foundry consists of the following tasks:


Collect Pre-installation Data
Install, Configure, and Deploy Universal Messaging (optional)
Install Apama for Cloud Foundry
Configure Apama for Deployment
Configure the Apama Security Manager (optional)
Define Service Plans (optional)
Deploy Apama to Cloud Foundry via BOSH
Create and Add the Apama Buildpack to Cloud Foundry (optional)

1. Collect Apama configuration data required for deployment and settings you provide to the Cloud Foundry operations team. See "Collecting Pre-installation Data" on page 11.
2. (Optional) Install, configure, and deploy Universal Messaging to the Cloud Foundry environment using BOSH. Create a Universal Messaging service using the boshDeploy script. See *Universal Messaging on Cloud Foundry* .

   Note: Perform this task only if you use the Apama with Universal Messaging deployment model. See "Deploying Apama to Cloud Foundry using BOSH" on page 35.

3. Install Apama on a bastion server. The bastion server must have access to the Cloud Foundry environment, the BOSH command line interface (bosh CLI), and the Cloud Foundry command line interface (cf CLI). See "Installing Apama for Cloud Foundry" on page 23.

4. Configure the deployment, depending on the Apama deployment scenario. See *Connecting Apama Applications to External Components* or the relevant third-party documentation.
5. (Optional) Configure the Apama Security Manager to enforce restrictions on third-party Java projects. See "Configuring the Apama Security Manager" on page 25.
6. (Optional) Define the characteristics of the Apama service by customizing the service plans. See "Defining Service Plans" on page 29.
7. Deploy Apama to a Cloud Foundry environment using BOSH. See "Deploying Apama to Cloud Foundry using BOSH" on page 35.
8. (Optional) Create the Apama buildpack and add it to Cloud Foundry. See "Creating and Adding the Apama Buildpack to Cloud Foundry" on page 51.

3 Collecting Pre-installation Data

- Network Configuration 12
- BOSH Virtual Machine Configuration 12
- User Roles 15
- Cloud Foundry Security Groups 16
- Firewall Settings Requirements 16

The following topics explain how to collect Apama configuration data required for deployment and settings that you provide to the Cloud Foundry operations team.

Network Configuration

You must allocate a range of IP addresses and define this range in the deployment manifest for BOSH deployments. To allocate sufficient IP addresses for the BVMs, use the Classless Inter-Domain Routing (CIDR) block format in the following formula:

IP Range = CIDR Rounded (Service Broker + #Service Instance BVMs + Reserved IPs)

In this formula,

- *Service Broker* = 1
- *#Service Instance BVMs* is the number of BVMs to allocate for an Apama deployment.
- *Reserved IPs* = 7. The value depends on the version of BOSH and includes the gateway address.

You can use any CIDR calculator to compute the number of IP addresses you need. Software AG recommends allocating minimum 32 IP addresses (CIDR xxx.xxx.xxx.xxx/27) for a deployment, and optimally 128 (CIDR /25).

Important: If you deploy multiple Software AG products, Software AG recommends setting up a single network configuration to include all products.

BOSH Virtual Machine Configuration

You use the following types of BOSH virtual machines when deploying Apama:

- "Service Broker BVM" on page 12
- "Service Instance BVMs" on page 13
- "Compilation BVMs" on page 13

Service Broker BVM

The Service Broker BVM provides the implementation of the Cloud Foundry Service Broker V2 API as a REST service and interacts with the Service Instance Manager (SIM) to manage the Apama server instances (Correlators). The Service Broker BVM is specified as apama_broker in the deployment manifest.

The Service Broker determines which Service Instance BVM allocates an Apama server instance. Each deployment has only one Service Broker instance, because the BVMs are used for administration, not for high availability.

You can monitor the Service Broker by using the `monit` command in the BOSH virtual machine or by using the `bosh vms` command in the bosh CLI. If the Service Broker stops, it is automatically restarted by `monit`. If the automatic restart does not occur, you can restart the Service Broker by using the `monit` command inside the BVM or by using the `bosh start` command in the bosh CLI. The log files are written to the /var/vcap/sys/log directory and can be accessed using bosh CLI commands.

Service Instance BVMs

The Service Instance BVM contains the Apama server instance and a Service Instance Manager for managing this instance. If you choose a shared VM service plan, each Service Instance BVM runs a Service Instance Manager and several Apama server instances. The Service Instance BVM is specified as apama_server in the deployment manifest.

Because the input and processing speed of Apama determine its performance, you must allocate sufficient resources for the Apama Service Instance BVM. Use "Service Instance BVMs Configuration" on page 14, "Service Instance BVM in Apama with Direct Access Scenario" on page 18, "Service Instance BVM in Apama with a Connectivity Plug-in Scenario" on page 19, and "Service Instance BVM in Apama with Universal Messaging Scenario" on page 20 to adjust the resource settings according to your usage scenarios.

You can monitor the Service Instance by using the `monit` command in the BOSH virtual machine or by using the `bosh vms` command in the bosh CLI. If the Service Instance stops, it is automatically restarted by `monit`. If the automatic restart does not occur, you can restart the Service Instance by using the `monit` command inside the BVM or by using the `bosh start` command in the bosh CLI. The log files are written to the /var/vcap/sys/log directory and can be accessed using bosh CLI commands.

Compilation BVMs

Compilation BVMs are used during the deployment process to extract packages into the appropriate installation directory defined in the Cloud Foundry environment. This is required, because all components are pre-built binaries, such as jar files and executable images.

Service Broker BVM Configuration

The following table lists the minimum configuration parameters of the Service Broker BVM:

Resource	Value
CPU (cores)	1
Memory (RAM)	2 GB
OS Disk	8 GB

Resource	Value
Persistent Disk	10 GB
Non-persistent Disk	20 GB
Network IP Address	1 (static)
IP Port	8080

For firewall configuration settings, see "Firewall Settings for BVMs" on page 16.

Service Instance BVMs Configuration

The following table lists the minimum configuration parameters of the Service Instance BVMs:

Resource	Value
CPU (cores)	1
Memory (RAM)	8 GB
OS Disk	8 GB
Persistent Disk	100 GB
Non-persistent Disk	20 GB
Network IP Address	1 per BVM
IP Port (Instance Manager)	8080
Correlator Ports	15903-15903+Correlators
Correlators (per instance)	3 (default)

Note: You should adjust the CPU, memory, and persistent disk space values, depending on the expected load and usage patterns. If the number of Correlators is set higher or the load per Correlator is high, use the above configuration as initial settings.

For firewall configuration settings, see "Firewall Settings for BVMs" on page 16.

Compilation BVMs Configuration

The following table lists the minimum configuration parameters of the Compilation BVMs:

Resource	Value
CPU (cores)	1
Memory (RAM)	1 GB
OS Disk	8 GB
Workers	2
Re-usage	true

You do not provide any specific firewall settings for the Compilation BVMs.

User Roles

Three main user roles can access the components of Cloud Foundry to perform tasks during the deployment of Apama.

Installer of Software AG products to Cloud Foundry

The Installer role interacts with Cloud Foundry, Cloud Foundry BOSH, and Software AG products. Depending on the security requirements of the environment, the Installer role can act in a Deployer or a Registrant role.

- The Deployer role performs the Cloud Foundry BOSH deployment and must have administrative rights to perform the deployment via the bosh CLI utility.
- The Registrant role performs Cloud Foundry registration of the Service Broker and must have administrative rights to perform the registration via the cf CLI.

Important: If the Installer role is also responsible for diagnosing issues in the Apama instances, the role must have access to the components used by the DevOps role.

DevOps

The DevOps role creates and manages the Apama cloud foundry service and creates, deploys, or administrates cloud-native applications.

End User

The End User role has access to the cloud-native applications deployed to Cloud Foundry, but does not interact directly with the Cloud Foundry components.

For firewall settings for all user roles, see "Firewall Settings for User Roles " on page 21.

Cloud Foundry Security Groups

If the IP subnet of the BOSH deployment is not known to Cloud Foundry, you must configure a new Cloud Foundry security group or extend the definition of an existing group to include the IP subnet. The following example shows how to configure a security group, named 'softwareag'. In this example, the BOSH deployment is under the IP subnet '10.xx.xx.xx', which you can modify as required. Use the following code to define the security group:

```
[
           {
              "destination": "10.0.0.0-10.255.255.255",
              "protocol": "all"
           }
]
```

The following commands create a security group with the name 'softwareag' in the Cloud Foundry Organization 'Apama' and space 'Demo':

```
cf create-security-group softwareag softwareag-security-group.json
cf bind-security-group softwareag Apama Demo
```

Note: In the example commands, you bind to an organization and a space. Depending on your corporate security strategy, you can bind to a running application environment. Use the `'cf bind-running-security-group ..'` command to bind to a running application.

Firewall Settings Requirements

This section contains the firewall security settings for the Service Broker BVM, the Service Instance BVMs, and the three main user roles. The Compilation BVMs do not require any specific firewall settings.

Firewall Settings for BVMs

The following tables contain the firewall settings of the Service Broker BVM and the Service Instance BVMs. Firewall settings of the Service Instance BVMs differ, depending on the communication scenario in the Apama Cloud Foundry BOSH deployment. In the tables:

- Incoming is the named resource in the host column that initiates the connection and connects to this BVM on the specified port.
- Outgoing is the BVM that initiates the connection to the named resources in the host column and to the port on the named resource.
- If the Port column is empty, the port is standard or depends on the environment.

Service Broker BVM

Source	Host	Port	Comments
Incoming	CF Host URL	8080	Cloud Controller API uses the Service Broker REST API.
Incoming	Service Instances	8080	All Service Instance BVMs use the Service Broker REST API.
Outgoing	NATS	4222	Service Broker issues NATS messages for managing Cloud Foundry proxies. **Note:** Access to NATS is required only when `proxyed` is set to `true` in the deployment manifest. The default value of `proxyed` is `false`.
Outgoing	CF Host		Access to Cloud Foundry Host to issue calls to Cloud Controller API.
Outgoing	Proxy Host		Service Broker uses the Proxy Host, when this is required in the corporate security settings to access the Cloud Foundry Host or other hosts within the environment. **Note:** Access to NATS is required only when `proxyed` is set to `true` in the deployment manifest. The default value of `proxyed` is `false`.
Outgoing	DNS		Service Broker uses DNS lookups and routing information.
Outgoing	Service Instances	8080	Service Broker manages Apama server instances (Correlators) over the Service Instance REST API.

Source	Host	Port	Comments
Outgoing	Service Instances	15903-1590x	Service Broker configures Apama server instances (Correlators) over the Apama Admin APIs. The number of Correlators per instance is 1590x.

Service Instance BVM in Apama with Direct Access Scenario

Source	Host	Port	Comments
Incoming	Service Broker	8080	All Service Instance BVMs use the Service Broker REST API.
Outgoing	NATS	4222	Service Broker issues NATS messages for managing Cloud Foundry proxies. **Note:** Access to NATS is required only when `proxyed` is set to `true` in the deployment manifest. The default value is `false`.
Outgoing	Proxy Host		Service Broker works using the Proxy Host, when this is required in the corporate security setting to access the Cloud Foundry Host or other hosts within the environment. **Note:** Access to NATS is required only when `proxyed` is set to `true` in the deployment manifest. The default value is `false`.
Outgoing	DNS		Service Broker uses DNS lookups and routing information.
Outgoing	Service Broker	8080	Service Instances notify Service Broker of their state.
Incoming	CF Apps Runtime	15903-1590x	Cloud Foundry Application Runtime (Elastic Runtime) uses Apama direct APIs (engine_client API). **Note:** This is required only when direct access is allowed as a protocol in a service plan.

Source	Host	Port	Comments
Incoming	CF Apps Runtime	8080	Injection of Apama project into an Apama Correlator uses buildpack extension for Apama. **Note:** This is required only when you use buildpack extension for Apama in Cloud Foundry.

Service Instance BVM in Apama with a Connectivity Plug-in Scenario

Source	Host	Port	Comments
Incoming	Service Broker	8080	All Service Instance BVMs use the Service Broker REST API.
Outgoing	NATS	4222	Service Broker issues NATS messages for managing Cloud Foundry proxies. **Note:** Access to NATS is required only when `proxyed` is set to `true` in the deployment manifest. The default value is `false`.
Outgoing	Proxy Host		Service Broker works using the Proxy Host, when this is required in the corporate security setting to access the Cloud Foundry Host or other hosts within the environment. **Note:** Access to NATS is required only when `proxyed` is set to `true` in the deployment manifest. The default value is `false`.
Outgoing	DNS		Service Broker uses DNS lookups and routing information.
Outgoing	Service Broker	8080	Service Instances notify Service Broker of their state.
Incoming	CF Apps Runtime	8080	Injection of Apama project into an Apama Correlator uses buildpack extension for Apama. **Note:** This is required only when you use buildpack extension for Apama in Cloud Foundry.

Service Instance BVM in Apama with Universal Messaging Scenario

Source	Host	Port	Comments
Incoming	Service Broker	8080	All Service Instance BVMs use the Service Broker REST API.
Outgoing	NATS	4222	Service Broker issues NATS messages for managing Cloud Foundry proxies. **Note:** Access to NATS is required only when `proxyed` is set to `true` in the deployment manifest. The default value is `false`.
Outgoing	Proxy Host		Service Broker works using the Proxy Host, when this is required in the corporate security setting to access the Cloud Foundry Host or other hosts within the environment. **Note:** Access to NATS is required only when `proxyed` is set to `true` in the deployment manifest. The default value is `false`.
Outgoing	DNS		Service Broker uses DNS lookups and routing information.
Outgoing	Service Broker	8080	Service Instances notify Service Broker of their state.
Incoming	CF Apps Runtime	8080	Injection of Apama project into an Apama Correlator uses buildpack extension for Apama. **Note:** This is required only when you use buildpack extension for Apama in Cloud Foundry.
Outgoing	Universal Messaging Service Instances	9000-900x	Apama creates a Java Message Service (JMS) connection to a Universal Messaging Realm to publish or subscribe for messages.

Firewall Settings for User Roles

The following tables contain the firewall settings for the Installer, DevOps, and End User roles. In the tables:

- Source is the tool or application that must have access to the service in the Service column.
- Service is the service that requires access. The firewall must allow access to the machine on which the service is running.

Installer

Source	Service	Comments
BOSH CLI	NTP	Deployer role. The VM that runs the bosh CLI tool connects to the machine that runs NTP.
BOSH CLI	DNS	Deployer role. DNS lookup is done on the VM.
BOSH CLI (bosh)	BOSH Director	Deployer role. The VM that runs the bosh CLI tool connects to BOSH Director.
Cloud Foundry CLI (cf)	Cloud Controller	Registrant role. The VM that runs the cf CLI tool connects to the Cloud Controller API (CAPI) machine.
Cloud Foundry CLI (cf)	DNS	Registrant role. DNS lookup is done on the VM.

DevOps

Source	Service	Ports	Comments
All Tools	DNS	Standard	DNS lookup is done on the VM.
BOSH CLI (bosh)	BOSH Director	Standard	The VM that runs the bosh CLI tool connects to BOSH Director.

Source	Service	Ports	Comments
Cloud Foundry CLI (cf)	Cloud Controller	Standard	The VM that runs the cf CLI tool connects to the Cloud Controller API (CAPI) machine.
Dev Tools	CF Apps Runtime	Standard	The VM on which the tools are installed accesses the Cloud Foundry Application Runtime (Elastic Runtime).
Software AG Tools	Apama Correlator	15903-1590x	CLI tools provided by Apama for specialist purposes must have direct access to the Apama Correlator.
Injector for Apama Projects	Service Instance	8080	If you do not use the injector for Apama projects through a buildpack, the injector must have access through a Service Key to the Service Instance Manager.

End User

Source	Service	Comments
Client Apps	CF Apps Runtime	The VM that the client uses accesses the Cloud Foundry Application Runtime (Elastic Runtime).
Client Apps	DNS	DNS lookup is done on the VM.

4 Installing Apama for Cloud Foundry

Before you can deploy Apama for Cloud Foundry, you must install Apama on a bastion server. The bastion server must have access to the Cloud Foundry environment, bosh CLI, and Cloud Foundry CLI. For more information about the Apama installation procedure, see *Installing Apama*.

Important: When you install Apama using Software AG Installer, you must select Enablement for Cloud Foundry in the Apama node of the product selection tree.

You can select any directory for an Apama work directory. However, Software AG recommends keeping the Apama work directory in the Apama installation directory. To do this, in Software AG Installer, in the Configure Apama screen, do one of the following:

- If you keep the default Apama installation directory, enter `/opt/softwareag/Apamawork` in the Work directory field.
- If you change the default Apama installation directory, enter `<install-directory>/Apamawork` in the Work directory field.

Note: If you change the default Apama installation directory or the Apama work directory, you must modify the setup.properties file. See "Modifying the setup.properties File" on page 40.

5 Configuring the Apama Security Manager

You can provide Apama on Cloud Foundry as a service to third parties, who can inject their own Java projects. You must enforce certain restrictions to stop third parties from causing malfunction of the Apama Server Java Virtual Machine and prevent third-party projects from interfering with each other. The Apama server has an enabled Java Security Manager that you can use to block:

- Attempts to configure and use other security managers.
- Process create and process exit calls, or allow them based on the user configuration.
- All file access, except for the directories specified in the Apama Security Manager configuration file and all their sub-directories.
- All library access, except for the libraries in the directories specified in the Apama Security Manager configuration file and all their sub-directories.
- Access on localhost to the ports specified in the Apama Security Manager configuration file.

Using the Apama Security Manager Configuration File

Edit the jsm-config.properties file in the cf-Apama/templates/samples/bosh-lite directory and place it in the cf-Apama/templates/custom directory. The default configuration contains the following permissions that are necessary for most Apama projects:

```
allow-file-read=/etc,\
/tmp,\
/home/vcap,\
/dev/random,\
/dev/urandom,\
/var/vcap/packages/apama,\
/var/vcap/packages/apama_security_manager,\
/var/vcap/packages/java8,\
/var/vcap/packages/um,\
/var/vcap/packages/zementis,\
/var/vcap/data/packages/apama/common/ADAPA/lib
allow-file-write=/tmp,\
/home/vcap
block-ports=
allow-process-create=false
allow-process-exit=false
allow-libraries=/var/vcap/packages/apama/lib
```

Software AG does not recommend removing entries from the default configuration.

Permissions of the Apama Security Manager Configuration File

You can configure the following permissions:

Permission	Allow All	Block All	Description
`allow-file-read`	/	Leave empty or remove permission	Directories where read file access is allowed.
`allow-file-write`	/	Leave empty or remove permission	Directories where write file access is allowed.
`allow-file-execute`	/	Leave empty or remove permission	Directories where execute file access is allowed. Default: empty.
`block-ports`	Leave empty or remove permission	/	List of blocked ports. For example: ■ `12345-13456` blocks all ports in this range ■ `-10000` blocks all ports lower than 10000 ■ `15000-` blocks all ports higher than 15000 ■ `14000` blocks port 14000 **Note:** The correlator ports and all ports higher than the highest correlator port plus 50 are always blocked.
`allow-process-create`	`true`	`false`	Allow the application to start a new process. **Note:** If you set the value to `true`, the directories under `allow-file-execute` are also checked to determine if execution of the particular file is allowed.

Permission	Allow All	Block All	Description
`allow-process-exit`	`true`	`false`	Allow the application to terminate the process.
`allow-libraries`	/	Leave empty or remove permission	Load libraries only from the allowed directories.

6 Defining Service Plans

■ Default Service Plans 30
■ Using the Service Plan Script 30

Cloud Foundry service plans define the characteristics of an Apama service. You can add, delete, or modify service plans by using additional parameters that control the service characteristics.

Service providers can create as many services as required. The number of service plans is not limited by the implementation.

Note: Apama on Cloud Foundry does not support Event Processing Language (EPL) plug-ins and connectivity plug-ins that are not written in Java.

Default Service Plans

The following pre-configured service plans are available:

Supported Feature	Free Plan	Bronze Plan	Silver Plan	Gold Plan
JMS protocol (Universal Messaging)	Yes	Yes	Yes	Yes
Connectivity plug-ins (Java)	No	Yes	Yes	Yes
Predictive analytics	No	No	No	Yes
EPL persistence	No	Yes	No	Yes
Dedicated VM plan	No	No	Yes	Yes
Direct communication	No	No	No	Yes

Important: To change the parameters of a service plan with a particular name, delete the plan and create a new plan using the same name and the new parameters.

Using the Service Plan Script

Creating a Custom Plan

You can create a custom plan, using the script named `plan` in the cloudfoundry/cf-Apama/cf-Apama/scripts directory. In the script, you must provide:

- A unique name for the service plan that consists of lowercase characters and no white spaces.
- The service plan description.

The script generates the service plan in the cloudfoundry/cf-Apama/cf-Apama/src/ plans directory. You can modify the service plan to specify and configure the custom characteristics required for this plan.

Note: The `runtimeProperties` section of the plan is a YAML representation of the plan metadata fields of Cloud Foundry's Catalog Metadata.

You can use properties in the plan metadata fields with the plan. You can add cost information based on the definition in the plan metadata field. For more details, see the Cloud Foundry documentation at https://docs.cloudfoundry.org/.

To deploy a service plan, you must deploy Apama to Cloud Foundry, using the boshDeploy script.

Custom Properties of the Service Plan

You can configure the following custom properties:

Property	Type	Default	Description

`protocols`	List of Strings (1..n)	Empty list	Defines the Apama protocols that you can enable for a plan. The `protocols` property is a list of protocols that are enabled or configured on a Correlator. By default, no protocols are enabled. You can specify the following values for the protocols:
			`protocols:`
			`  - jms`
			`  - connectivity`
			`  - direct`
			When you provide one of the properties, the value limits the information returned by a service binding or a service key.

`dedicated`	Boolean	false	When set to `true`, the service plan exclusively allocates a Correlator to a BVM. When set to `false`, the service plan does not exclusively allocate a Correlator to a BVM.
`persistence`	Boolean	false	When set to `true`, the service plan enables EPL persistence support for the Apama Correlator.

Property	Type	Default	Description
			When set to `false`, the service plan does not enable EPL persistence support for the Apama Correlator.

`adapters`	List of Strings (1..n)	Empty list	Enables adapters for Apama. If you specify:
			`adapters:`
			`  - predictiveAnalytics`
			Apama enables the Predictive Analytics adapter. If you enable the Predictive Analytics adapter, the Apama project must contain a *.pml file with configuration settings for Predictive Analytics.

Parameters of the Service Plan Script

You can configure the following parameters of the service plan script:

Parameter	Description
`-n | --name <name>`	Required. Unique name for the service plan that consists of lowercase characters and no white spaces.
`-d | --description <description>`	Service plan description. If the description consists of more than one word, you must enclose the phrase in quotes.
`-r | --remove`	Remove the service plan. **Important:** This parameter removes the service plan. You cannot retrieve or restore a removed service plan.
`-f | --force`	Overwrite the service plan and reset it to the initial template values.
`-h | --help`	Display the service plan script help.

How to Execute the Service Plan Script on Linux

1. Go to <install-directory>/cloudfoundry/Apama/cf-Apama/scripts.

2. Execute the ./plan script by specifying a name and a description for the new plan. For example, you can create a service plan named 'platinum':

   ```
   cfdev:/opt/softwareag/cloudfoundry/apama/cf-Apama/
   scripts$ ./plan -n "platinum" -d "Premier Apama plan containing
   all super features"
   ```

   The command output gives the location of the file.

   ```
   Created plan: /opt/softwareag/cloudfoundry/apama/
   cf-Apama/scripts/../src/plans/platinum.yml
   Edit then redeploy using boshDeploy -r -c
   ```

   Note: Follow the requirements for name and description in "Parameters of the Service Plan Script" on page 32.

3. Edit the `runtimeProperties` section of the generated plan file as required. For example:

   ```
   ---
   id: "ccc61e30-428f-4430-a808-aef6171836b8"
   name: "platinum"
   description: "Premier Apama plan containing all super features"
   free: true
   runtimeProperties:
     dedicated: false
   #  persistence: true
   #  adapters:
   #    - predictiveAnalytics
   #  protocols:
   #    - jms
   #    - connectivity
   #    - direct
   ```

4. Deploy the Apama Service to the Cloud Foundry Bosh environment using the boshDeploy script with the `-r` and `-c` parameters.

Considerations for the Service Plan Script

Consider the following when you work with the service plan script:

- Because each plan must have a unique `id` attribute in the JSON file, do not copy files in the plans directory. Use the service plan script to create new plans.
- If you use a plan and modify this plan afterwards, the plan modifications affect only the services that you create after deployment.
- If you modify a plan and redeploy after the initial deployment, use the Cloud Foundry `cf update-service-broker` command to update Cloud Foundry's definition of the service broker. For example:

  ```
  cf update-service-broker apama admin admin http://10.245.0.136:8080
  ```

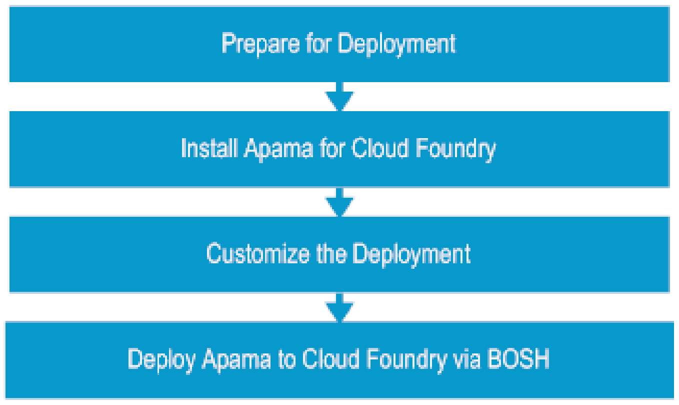
7 Deploying Apama to Cloud Foundry using BOSH

- Before You Begin 40
- Customizing Your Deployment 40
- Deploying Apama Using the boshDeploy Script 50

You can deploy Apama to a Cloud Foundry environment by deploying the Apama server instance on a set of BVMs. An Apama deployment to Cloud Foundry consists of the following two jobs or instance groups:

- Apama Service Broker, which implements the Cloud Foundry Service Broker V2 API. The Apama Service Broker is deployed on one Cloud Foundry BVM. The Service Broker BVM is specified as apama_broker in the deployment manifest.
- Apama server, which contains an Apama service instance. The Apama server is deployed to another BVM. The Service Instance BVM is specified as apama_server in the deployment manifest.

To deploy Apama for Cloud Foundry using BOSH, you must complete the following tasks:

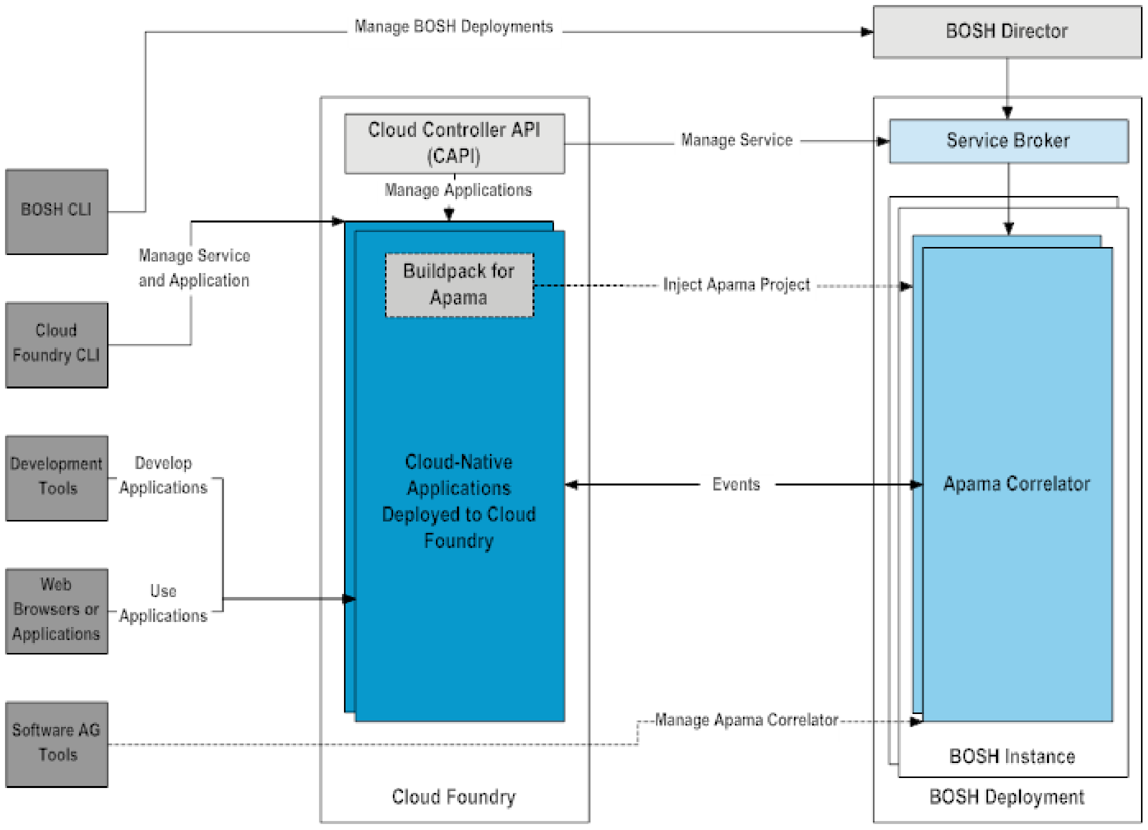

Prepare for Deployment
Install Apama for Cloud Foundry
Customize the Deployment
Deploy Apama to Cloud Foundry via BOSH

1. Install the latest versions of tools required for the deployment. See "Before You Begin" on page 40.
2. Install Apama on a bastion server. The bastion server must have access to the Cloud Foundry environment, bosh CLI, and Cloud Foundry CLI. See "Installing Apama for Cloud Foundry" on page 23.
3. Customize your deployment by modifying the settings.properties, cloud-config.yml, cf-Apama.yml, custom-vars.yml, and creds.properties files. See "Customizing Your Deployment" on page 40.
4. Deploy Apama to a Cloud Foundry environment using BOSH. See "Deploying Apama Using the boshDeploy Script" on page 50.

The following sections show the applications, components, tools, and communication paths in the three Apama deployment configurations.

Deploying Apama with Direct Access for Cloud Foundry Using BOSH

In the direct access scenario, communication goes over the Apama client API interface (engine_client API). You can use the engine_client API to connect directly to the Apama Correlator and to send events in their native form.



Important: Because the direct access scenario does not have any authentication to restrict who can send events, Software AG recommends using firewall and network level security of Cloud Foundry to restrict access to the Apama Correlator port.

Deploying Apama with a Connectivity Plug-in for Cloud Foundry Using BOSH

Connectivity plug-ins enable custom communications, filtering, and transformation mechanisms in Apama. When you use Apama with a connectivity plug-in, you must enable the use of connectivity plug-ins in the Apama service plan.

```
runtimeProperties:
  protocols:
   - connectivity
```

Because each connectivity plug-in has a different interface that depends on the implementation, the VCAP_SERVICES setting does not contain any details about the design of connectivity plug-ins. Your project must include a YAML file with the configuration of the connectivity plug-in that you use. All paths in the YAML file must be relative to the project root directory.

Apama for Cloud Foundry supports only connectivity plug-ins that are written in Java. You can use the Java plug-ins that are included with Apama or use your own Java plug-ins. For more information how to include your own plug-ins in the Apama project, see *Connecting Apama Applications to External Components*.

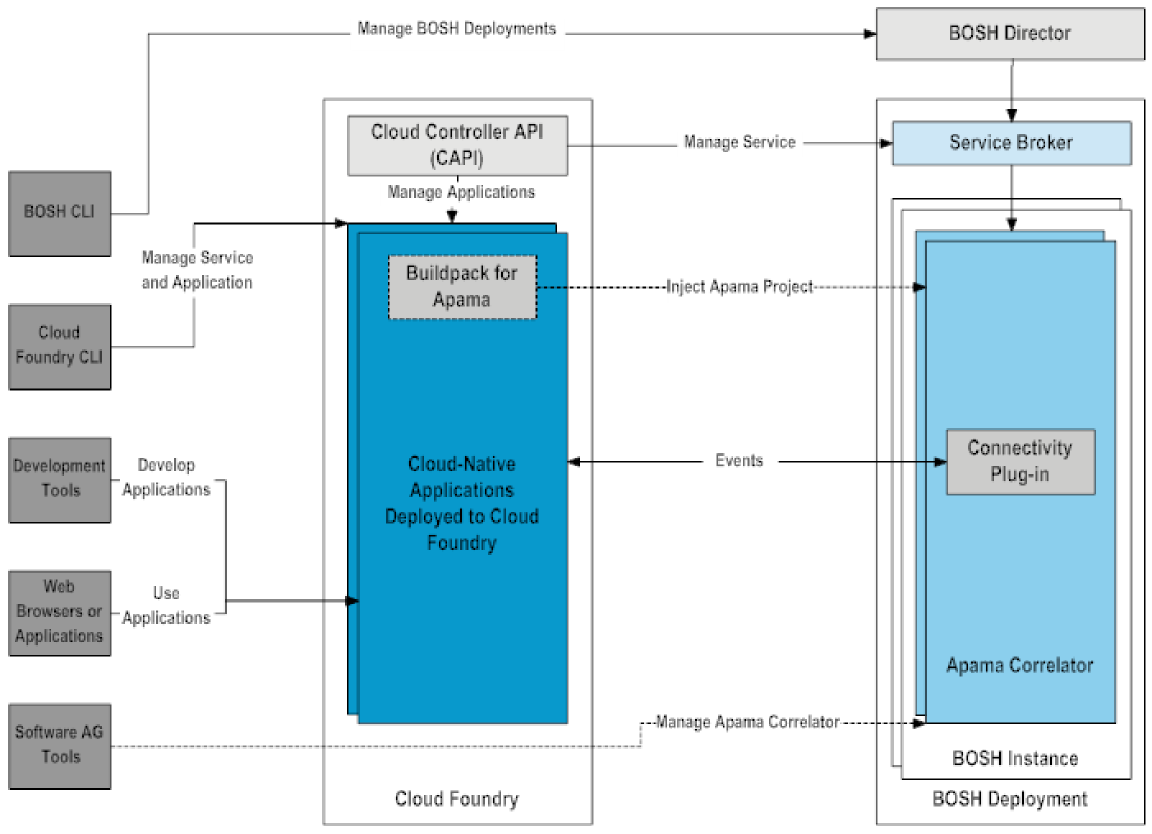
To verify that a connectivity plug-in is written in Java, ensure that the `classpath` property in the connectivity plug-in configuration has a .jar extension. For example, Apama for Cloud Foundry supports the following connectivity plug-in configuration:

```
connectivityPlugins:
  <customPlugin>:
    directory: ${APAMA_WORK}/inject/<customPluginDir>
    classpath: <customPluginJar>.jar
    class: <CustomPluginClass>
  jsonCodec:
    directory: ${APAMA_HOME}/lib/
    classpath: json-codec.jar
    class: com.softwareag.connectivity.plugins.JSONCodec
```

Apama for Cloud Foundry does not support the following connectivity plug-ins:

- Universal Messaging transport
- Digital Event Services transport

Important: Port 8080 is reserved for management services. Do not use this port for your project. Verify that Cloud Foundry or the firewall do not block any other ports that you need for your work with the project.



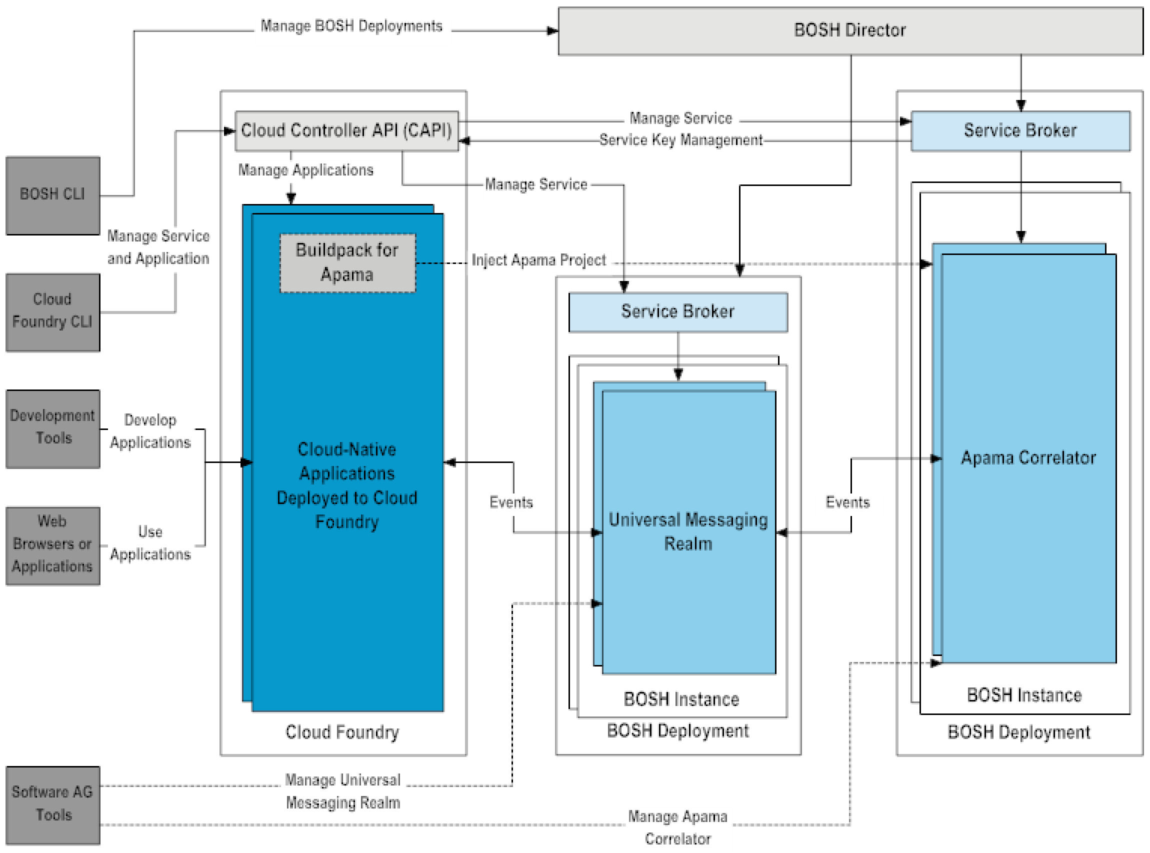
For more information about connectivity plug-ins, see *Introduction to Apama* and *Connecting Apama Applications to External Components*.

Deploying Apama with Universal Messaging for Cloud Foundry using BOSH

The Apama with Universal Messaging scenario is the default method of sending and receiving events. When you use Apama with Universal Messaging, you must enable JMS communication in the service plans of both Apama and Universal Messaging.

```
runtimeProperties:
  protocols:
   - JMS
```

Because the Cloud Foundry project communicates only with Universal Messaging in the Apama with Universal Messaging scenario, ensure that your project contains the UniversalMessaging-spring.properties and UniversalMessaging-spring.xml files. The UniversalMessaging-spring.properties and UniversalMessaging-spring.xml files are created automatically when you create an Apama with Universal Messaging project in Software AG Designer. Ensure that the configuration settings in the project are the same as the configuration settings in the Universal Messaging deployment.



For information about installing, configuring, and deploying Universal Messaging to Cloud Foundry, see *Universal Messaging on Cloud Foundry*. For information about creating an Apama project, see *Using Apama with Software AG Designer*.

Before You Begin

Before you start the deployment of Apama to Cloud Foundry, you must download and install the latest versions of the following tools:

- To interact with your infrastructure provider and perform actions in the cloud, install bosh CLI. See https://bosh.io.
- To execute Cloud Foundry commands, install Cloud Foundry CLI (cf CLI). See https://github.com/cloudfoundry/cli.
- To use as a YAML merging tool and create deployment manifests, install Spruce. See https://github.com/geofffranks/spruce/releases.
- To use a JSON querying tool, install JQ. See https://stedolan.github.io/jq/download.

Customizing Your Deployment

Because every Cloud Foundry environment is unique, you must provide details about your Cloud Foundry deployment environment in the Apama deployment manifest. Store your custom deployment files in the cf-Apama/templates/custom directory. Note that updates to Cloud Foundry do not overwrite the custom configuration files.

Before you create a custom configuration, you must create a deployment manifest. For more information, see https://bosh.io.

Use the following steps to create a custom deployment:

1. In the cf-Apama/templates/custom directory, modify the setup.properties file to provide the path to the Apama installation on the machine and to the Apama work directory. See "Modifying the setup.properties File" on page 40.
2. Create custom YAML files for the deployment. See "Customizing the YAML Files" on page 41.
3. Create additional customizations by using custom operation files and BOSH links. See "Creating Additional Customizations" on page 48.
4. Create a creds.properties file for the creation of Passman encrypted passwords. See "Customizing the creds.properties File" on page 49.

Modifying the setup.properties File

To provide the path to the Apama installation on the machine and to the Apama work directory, you must modify the setup.properties file in the cf-Apama/templates/custom directory. For details, see the table below.

Apama installation directory	Apama work directory	setup.properties File

Default	Default	Do not modify the setup.properties file. The default entry is:
		`# Installation directory`
		`# for Software AG products (Apama)`
		`export INSTALLDIR=/opt/softwareag`
		`# Default path to the Apamawork directory`
		`export APAMAWORKDIR=${INSTALLDIR}/Apamawork`

Custom	Default	Edit only INSTALLDIR in the setup.properties file. For example:
		`# Installation directory`
		`# for Software AG products (Apama)`
		`export INSTALLDIR=/home/apama/softwareag`
		`# Default path to the Apamawork directory`
		`export APAMAWORKDIR=${INSTALLDIR}/Apamawork`

Custom	Custom	Edit INSTALLDIR and APAMAWORKDIR in the setup.properties file. For example:
		`# Installation directory`
		`# for Software AG products (Apama)`
		`export INSTALLDIR=/home/apama/softwareag`
		`# Default path to the Apamawork directory`
		`export APAMAWORKDIR=/home/apama/Apamawork`

Customizing the YAML Files

Software AG recommends that you create the following YAML files and place them in the custom directory:

- A cloud-config.yml file that contains details about the BVMs and the Infrastructure as a Service (IaaS) specific settings of the Cloud Foundry environment.
- A cf-Apama.yml file that contains configuration settings for the Apama deployment.
- A custom-vars.yml file that contains the definitions for all variables that are implicitly declared in the cf-Apama.yml file or used in the cloud-config.yml file.

Multiple configuration files simplify the configuration of multiple environments. By creating separate YAML configuration files, you separate the IaaS-specific configuration from the deployment configuration. For example, you can use the same cloud-config.yml file for test, pre-production, and production environments.

You can modify the sample YAML configuration files for BOSH-Lite in the cf-Apama/templates/samples/bosh-lite directory as required.

cloud-config.yml

The cloud-config.yml file contains the IaaS-specific settings of the Cloud Foundry environment. The sample BOSH-Lite configuration in the following sections provides details and examples about setting up the configuration files for the environment. You can edit the cloud-config.yml file in any editor that supports YAML validation. Depending on your IaaS environment, you should define additional configuration sections.

Availability Zones

```
apama_broker:
  instances: 1
  azs:
  - z1
apama_server:
  instances: 1
  azs:
  - z1
```

To specify in what availability zone to deploy the Service Broker BVM and the Service Instance BVM, configure the `azs` property of `apama_broker` and `apama_server`. Software AG recommends that you specify one availability zone for the Service Broker BVM and the Service Instance BVM.

For more information about the Service Broker BVM and the Service Instance BVMs, see "BOSH Virtual Machine Configuration" on page 12.

Disk Types

```
disk_types:
- disk_size: 5120
  name: 5GB
- disk_size: 10240
  name: 10GB
```

Software AG recommends that you use two disk types to deploy Apama:

- One disk type for the Apama Service Broker. Because the Apama Service Broker does not store large data, this disk type does not require as much space as the disk type for the Apama Correlators.
- One disk type for the Apama Correlators. Persistent files that Apama requires when recovering from a failure are stored in this disk type. You must configure a type that is large enough to handle the information. For information about storage, see *Introduction to Apama*. For disk type configuration, see "Collecting Pre-installation Data" on page 11.

The names of the persistent disks must remain unchanged, because they are referenced in other parts of the manifest.

You can define a set of cloud property values based on your IaaS environment to allocate the disk space on a Storage Area Network (SAN).

Networks

```
networks:
- name: apama-net
  subnets:
  - azs: [z1, z2, z3]
    cloud_properties:
      name: ((apama-network))
    gateway: 10.246.0.1
    range: 10.246.0.0/20
    dns: ((internal_dns))
    reserved:
    - 10.246.0.1 - 10.246.0.6
    static: 10.246.0.7
  type: manual
```

The network topology of your Cloud Foundry deployment is specific to your environment and to your IaaS infrastructure. For an Apama deployment:

- Because the `name` and `network` values are referenced in other sections of the manifest, ensure that the values are consistent.
- In the static IP subsection, specify at least one static IP address for the Apama Service Broker.
- All IP addresses for the Apama Service are dynamically allocated by BOSH at deployment time.

You can also add the following properties in your environment:

- IP address of the DNS server.
- Cloud specific properties for the network virtualization layer.

You should have a range of IP addresses for deployment that is sufficient for the initial deployment. Software AG recommends that you use a larger IP range in order to scale the deployment to more BVMs if necessary.

Compilation

```
compilation:
  az: z1
  network: default
  reuse_compilation_vms: true
  vm_type: m3.medium
  workers: 4
```

The compilation BVMs are used by BOSH during deployment. Ensure that the network name is consistent across the manifest. If you increase the number of `workers`, you might deploy faster, but you allocate more BVMs. Usually, these BVMs are temporary and are created when required. This has a negative effect on your environment or leads to additional costs for IaaS services. Software AG recommends that you use two workers.

Depending on the IaaS environment, you might need a set of cloud properties for the compilation BVMs.

cf-Apama.yml

The cf-Apama.yml configuration file contains the parameters that are necessary for the Apama BOSH deployment. You can edit the cf-Apama.yml file in any editor that supports YAML validation.

Parameters for Apama Service Broker (apama_broker)

Parameter	Required	Description
`proxy_base`	yes	Host of the Cloud Foundry environment. When creating proxy URLs, the host is of type <serviceid>.<proxy_base>.
`proxyed`	yes	When set to `true`, proxies are generated to access services outside of Cloud Foundry. When set to `false`, proxies are not generated to access services outside of Cloud Foundry. Default: `false`
`port`	yes	The port used by the Apama Service Broker REST API, which implements the Cloud Foundry Service Broker V2 API.
`cf_apiurl`	yes	Cloud Foundry API V2 Endpoint URL. If you do not specify a URL as a manifest property, the default value is used. Default: `http://api.bosh-lite.com`
`cf_admin`	yes	User that the Cloud Controller uses when talking to the Service Broker. The password for this user is stored through Passman using the creds.properties file. The username/password combination must match the combination given during service registration in Cloud Foundry.
`broker_admin`	yes	Username used for communication between the Service Broker and the Service Instance Managers. The password for this user is stored through Passman using the cred.properties file.
`nathost`	no	NAT Host IP address.

Parameter	Required	Description
		Note: Access to NATS is required only when `proxyed` is set to `true` in the deployment manifest. The default value of `proxyed` is `false`.
`natuser`	no	Username for accessing NAT service. The password is stored encrypted. See the creds.properties file. **Note:** Access to NATS is required only when `proxyed` is set to `true` in the deployment manifest. The default value of `proxyed` is `false`.
`check_cert`	yes	When set to `true`, certificate validation is performed when communicating with the OAuth server. When set to `false`, certificate validation is not performed when communicating with the OAuth server. Default: `false`
`capi_cert`	no	SSL key for establishing secured communication with Cloud Controller API (CAPI) and UAA OAuth server.
`async_threads`	no	The size of the thread pool used for asynchronous service creation. Change this property only if recommended by Software AG. Minimum: `1` Maximum: `50` Default: `5`
`cf_user`	no	Username, for which an OAuth2 token is generated. You use the OAuth2 token to request a key for a service of type 'Universal Messaging' from the CAPI.
`cf_authid`	no	Username, which is used for authentication against the User Account and Authentication (UAA) service. This username retrieves the

Parameter	Required	Description
		OAuth2 token that you need for service key retrieval.
`cf_clientid`	no	When you request an OAuth2 token from the UAA service, you must specify a value for `client_id`. If you do not specify a value, the value of `cf_authid` is used as `client_id`.
`Max_servers`	no	Defines the maximum number of correlators that you can allocate on a shared BVM. Default: `3`
`certpassword`	no	Password to be used for certificate file. Default: `cfc$C1oundF0undry`

Parameters for Apama Server (apama_server)

Parameter	Required	Description
`port`	yes	The port used by the Apama Service Instance Manager (SIM). The Apama Broker communicates with the SIM on this port.
`apamabroker`	yes	Do not edit. The IP of the Service Broker that is automatically computed when you use the boshDeploy script.
`apamabrokerport`	yes	Do not edit. The port number of the Service Broker that is automatically computed when you use the boshDeploy script.
`apamabrokerprotocol`	yes	Do not edit. The communication protocol used to contact the Service Broker.
`cf_admin`	yes	User that the Cloud Controller uses when talking to the Service Broker. The password for this user is stored through Passman using the creds.properties file. The username/password combination must match the combination given during service registration in Cloud Foundry.

Parameter	Required	Description
		Specify the same value as the value of the Service Broker's `cf_admin` parameter.
`broker_admin`	yes	Username used by all SIMs to validate incoming requests. The password is stored through Passman. Specify the same value as the value of the Service Broker's `broker_admin` parameter.
`correlator_ports`	yes	Defines the range of correllator ports that the correlator uses when `direct` is enabled. When `direct` is disabled, the ports are outside of this range. The ports in this range must be accessible outside of the BVM, for example in Cloud Foundry Elastic runtime. Default: `15903-15925`
`inject_timeout`	no	The maximum amount of time in seconds to wait for the Apama project injection to finish. Default: `3600`

Update

```
update:
  canaries: 1
  canary_watch_time: 15000-30000
  max_in_flight: 1
  update_watch_time: 15000-300000
```

The `update` section is used for redeploying of services and defining how BOSH must roll out updates. Contact your Cloud Foundry administrator to check if other parameters are necessary in your environment.

custom-vars.yml

Bosh CLI provides a special syntax in YAML documents to annotate that some properties are set externally. You can use this syntax to convert the YAML documents into parameterized templates.

All properties in the cf-Apama.yml file are in the format:

```
property: ((variable_name))
```

All custom variables are placed in the custom-vars.yml file in the custom directory and can be:

- Global variables that are used by both apama_broker and apama_server jobs.
- Global variables in the cf-Apama.yml file.

- Specific variables for the apama_broker job.
- Specific variables for the apama_server job.

For example, you can have the following variables in the custom-vars.yml file:

```
# global vars
stemcell_version: latest
apama_broker:
  instances: 1
  ip_address: 10.246.0.7
  port: 8080
apama_server:
  instances: 1
  port: 8080
```

For more information about variables, see https://bosh.io.

Creating Additional Customizations

You can additionally customize your Apama deployment by using custom operation files and BOSH links.

Custom Operation Files

To simplify fix installations and upgrades, you can use an operation file, named custom-ops.yml, to modify deployment manifest files. Using the operation file, you can update, insert, or delete elements in the manifest files, as required.

You create and edit the custom-ops.yml file in the cf-Apama/templates/custom directory. Entries in the custom-ops.yml file can overwrite properties in the main deployment manifest or add new properties specific to your requirements.

BOSH Links

You can specify links between jobs in bosh CLI manifests, so that certain jobs can receive information about other jobs. By using links in your manifests, you can remove unnecessary variables and avoid issues if jobs properties change. Bosh CLI automatically detects the dependencies and restarts or updates the dependent jobs.

Important: Note that you can use links to refer to apama_broker properties in apama_server. If some of the referenced properties change, both apama_broker and apama_server instances are redeployed or updated.

To use a BOSH link, do the following:

- Include provider or consumer references in the deployment manifest file.
- In the job spec files, declare that the jobs provide or consume information from other jobs.
- Remove variables and references from the job entries.
- Update the job template (.erb) files to use link references to get the properties from other jobs.

For example, in the Apama deployment manifest, the apama_broker instance group provides the port and protocol information through a link that the apama_server instance group consumes.

```
# cf-Apama.yml
name: cf-apama
instance_groups:
- name: apama_broker
  jobs:
  - name: apama_broker
    provides:
      apama_broker:
        as: apama_broker_conn
# ...
- name: apama_server
  jobs:
  - name: apama_server
    consumes:
      apama_broker:
        from: apama_broker_conn
# ...
```

The information is declared in the spec file of apama_broker.

```
# apama_broker spec file
name: apama_broker
provides:
- name: apama_broker
  type: conn
  properties:
  - port
  - protocol
```

The spec file of apama_server declares that it consumes this information.

```
# apama_server spec file
name: apama_server
consumes:
- name: apama_broker
  type: conn
```

In the server.properties.erb file of the apama_server job, the information is included in the link.

```
# jobs/apama_server/templates/server.properties.erb
BROKER=<%= link('apama_broker').p('protocol') %>
://<%= link('apama_broker').instances[0].address %>
:<%= link('apama_broker').p('port') %>/broker
```

Customizing the creds.properties File

The creds.properties file is an input file to create a Passman encrypted password file that is deployed to Service Broker and Service Instance Manager nodes. This file is a simple name-value pair file in the format username=password. The usernames in the creds.properties file must match the usernames specified as values of the following parameters in the cf-Apama.yml file:

- `cf_authid`
- `cf_clientid`

- `broker_admin`
- `cf_user`
- `cf_admin`
- `natuser`

Follow your corporate rules when you create passwords. Software AG recommends choosing a unique username for each of these users. If you use a specific username and password for more than one parameter, you must define this username and password only once in the creds.properties fie.

Deploying Apama Using the boshDeploy Script

After you configure your environment, use the boshDeploy script located in the cf-Apama/scripts directory to perform any of the following deployment scenarios:

Deployment scenario	To perform this scenario...
Deploy to Cloud Foundry BOSH and register with Cloud Controller.	Run the boshDeploy script without any parameters.
Deploy as two separate calls to the deployment script. **Note:** Use this option to deploy in secure environments, where the roles of deploying to Cloud Foundry BOSH and registering with Cloud Foundry are separated.	1. Deploy to Cloud Foundry BOSH by running the boshDeploy script with the `-n | --nocf` parameter. 2. Register with Cloud Controller by running the boshDeploy script with the `-c | --cfonly` parameter.
Redeploy to Cloud Foundry BOSH and register with Cloud Controller when the Apama deployment has no running service instances.	Run the boshDeploy script without any parameters.
Redeploy to Cloud Foundry BOSH and register with Cloud Controller when the Apama deployment has running service instances.	Run the boshDeploy script with the `-r | --redeploy` parameter.

Important: All deployment scenarios, except running the boshDeploy script with the `-r | --redeploy` parameter, delete the existing BOSH deployment.

8 Creating and Adding the Apama Buildpack to Cloud Foundry

Before you create the Apama buildpack, download and install the following tools:

- To provide an environment for Ruby projects, install Bundler. See http://bundler.io/.
- To use as a build utility for Ruby, install rake. See https://github.com/ruby/rake.

The scripts for creating the Apama buildpack are located in the /opt/softwareag/cloudfoundry/Apama/cf-Apama/buildpack/java directory. Use the createBuildpack script to create the buildpack:

```
$ cd /opt/softwareag/cloudfoundry/Apama/cf-Apama/buildpack/java
$ ./createBuildpack
```

The createBuildpack script creates an offline Apama buildpack in the java-buildpack/build directory. The Apama buildpack automatically uploads to Cloud Foundry at position 1.

```
[apama@vmbgcloudsandbox01 ~]$ cf buildpacks
Getting buildpacks...
buildpack                     position   enabled   locked   filename
sag-offline-java-buildpack    1          true      false    java-buildpack
                                                            -offline-cf61c4c.zip
staticfile_buildpack          2          true      false    staticfile_buildpack
                                                            -cached-v1.3.14.zip
java_buildpack                3          true      false    java-buildpack
                                                            -v3.10.zip
ruby_buildpack                4          true      false    ruby_buildpack-cached
                                                            -v1.6.29.zip
nodejs_buildpack              5          true      false    nodejs_buildpack
                                                            -cached-v1.5.24.zip
go_buildpack                  6          true      false    go_buildpack-cached
                                                            -v1.7.16.zip
python_buildpack              7          true      false    python_buildpack
                                                            -cached-v1.5.13.zip
php_buildpack                 8          true      false    php_buildpack-cached
                                                            -v4.3.23.zip
binary_buildpack              9          true      false    binary_buildpack
                                                            -cached-v1.0.5.zip
dotnet_core_buildpack         10         true      false    dotnet-core_buildpack
                                                            -cached-v1.0.6.zip
```

9 Installing Apama Fixes

You can install the latest fixes of Apama on a running Apama deployment in Cloud Foundry by using the Software AG Update Manager.

To install Apama fixes on Cloud Foundry

1. Install the fix of Apama on the bastion server. For more information on installing fixes, see *Using the Software AG Update Manager*.
2. Deploy the new version of Apama on Cloud Foundry by using the boshDeploy script with an additional `-r` parameter. For more information on the boshDeploy script, see "Deploying Apama Using the boshDeploy Script" on page 50.

   Note: You will lose all data received at the time of redeployment of Apama on Cloud Foundry, because Cloud Foundry BOSH restarts all running services.

Example of Fix Installation

```
<Install the fix according to the "Using the Software AG Update Manager" guide.>
<installdir>/cloudfoundry/Apama/cf-Apama/scripts/boshDeploy -r
```

10 Upgrading Apama on Cloud Foundry

To upgrade Apama on Cloud Foundry

1. Back up the customized configuration files of the current Apama on Cloud Foundry installation.
    a. Back up the following files in the <installdir>/cloudfoundry/Apama/cf-Apama/templates/custom directory:
        - creds.properties
        - jobs.yml
        - log4j-debug.properties
        - log4j.properties
        - setup.properties

    b. Back up the deployment manifest file cf-Apama.yml in the <installdir>/cloudfoundry/Apama/cf-Apama/deployment directory.
2. Upgrade the Apama installation on the bastion server. For more information on the upgrade, see *Upgrading Software AG Products*.
3. In the new Apama on Cloud Foundry installation directory, copy the following configuration files from the <installdir>/cloudfoundry/Apama/cf-Apama/templates/samples/bosh-lite directory to the <installdir>/cloudfoundry/Apama/cf-Apama/templates/custom directory:
    - creds.properties
    - custom-vars.yml
    - jsm-config.properties
    - log4j-debug.properties
    - log4j.properties
4. Follow the guidelines to transfer the changes from the customized configuration files that you back up in step 1 to the corresponding configuration files in the new Apama on Cloud Foundry 10.1 installation.
    a. If you overwrite the creds.properties file, add the `softwareag=sag` property.

    b. If you overwrite the setup.properties file, add the following variables:

    ```
    # Default path to the Apamawork directory
    export APAMAWORKDIR=${INSTALLDIR}/Apamawork
    # Name of BOSH CLI tool
    export BOSHCLI=bosh2
    ```

For information about the Apama work directory, see "Modifying the setup.properties File" on page 40.

c. Transfer the customized information from jobs.yml and vm-config.yml in Apama on Cloud Foundry 10.0 to custom-vars.yml and cf-Apama.yml in Apama on Cloud Foundry 10.1.

For information about the configuration and manifest files, see "Customizing Your Deployment" on page 40.

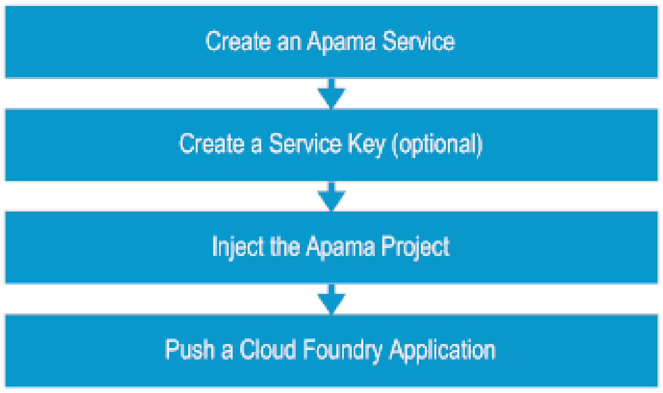
5. Deploy the new version of Apama on Cloud Foundry by using the boshDeploy script with an additional `-r` parameter. For more information on the boshDeploy script, see "Deploying Apama Using the boshDeploy Script" on page 50.

Note: You will lose all data received at the time of redeployment of Apama on Cloud Foundry, because Cloud Foundry BOSH restarts all running services.

11 Using Apama on Cloud Foundry

- Creating an Apama Service 58
- Creating a Service Key 59
- Client Access to the Apama Service 60
- Injecting the Apama Project 61
- Collecting Diagnostic Information 65

The process of using Apama on Cloud Foundry consists of the following tasks:


Create an Apama Service
Create a Service Key (optional)
Inject the Apama Project
Push a Cloud Foundry Application

1. Create an Apama cloud foundry service. See "Creating an Apama Service" on page 58.
2. (Optional) Create a service key. See "Creating a Service Key" on page 59.
3. Inject the Apama project. See "Injecting the Apama Project" on page 61.
4. In the cf CLI, push a cloud foundry application that uses the Apama cloud foundry service.

Creating an Apama Service

Custom applications access Apama on Cloud Foundry through the Apama cloud foundry service. The DevOps user creates the Apama cloud foundry service after the deployment of Apama to Cloud Foundry.

To create an Apama service, use the createService script located in the cf-Apama/scripts directory. The script has the following parameters:

Parameter	Description
`-h | --help`	Display the createService script help.
`-b | --broker <name>`	Required. Specify the name of the Service Broker to use.
`-p | --plan <name>`	Required. Specify the name of the service plan to use.
`-s | --service <name>`	Required. Specify the name of the service to create.
`-c | --configuration <config>`	Specify a JSON string that contains the configuration of the service.

Parameter	Description
	Note: Use single quotes for attributes.
`-f | --file <filename>`	Specify a JSON configuration file.

For example, use the following command to create an Apama service named 'apama' that uses a previously created Universal Messaging service named 'umfree_1'.

```
scripts/createService -b apama -p free -s apama -c '{"UniversalMessaging":
"umfree_1"}'
```

Creating a Service Key

A service key contains the bindings that are generated by the Service Broker and expose ports for usage by Cloud Foundry applications. These bindings contain host and ports that should be exposed to the clients for communications. In a secure environment, applications might run either in a VMaaS environment or outside of Cloud Foundry Elastic Runtime. In such cases you can use a capability to get a binding entry that can be interpreted by firewall software to enable these connections in the non-Cloud Foundry Elastic Runtime environment.

You can use the service binding call in CAPI to pass properties through a JSON string. Use the `create-service-key` command in cf CLI to define these properties for each binding.

The service key properties `scope` and `firewall` define how an additional service binding entry is generated.

```
"scope": "firewall"
"firewall" : "<attribute name>"
```

These properties are passed as JSON attributes into the service binding extension mechanism. `<attribute name>` is the name of the binding entry that the firewall software understands. You must define a unique `<attribute name>` within the binding map. If you do not define the `scope` property, the `firewall` property is ignored.

For example, you can create a service key 'pam_firewall' for the service 'apama' by using the `scope` and `firewall` properties.

```
cf create-service-key apama pam_firewall
-c '{"scope":"firewall", "firewall":"serverlist"}'
cf service-key apama pam_firewall
{
"direct": "apama://10.246.0.8:15903",
"serverlist": "10.246.0.8:8080,10.246.0.8:15903",
...
 }
```

In this example you ask for a service key with an additional entry named "serverlist". This additional entry contains the host and ports to be exposed through the firewall for Apama.

Client Access to the Apama Service

Cloud Foundry applications that are bound to services such as Apama require credentials to connect to the service. These Cloud Foundry applications have an environment setting named VCAP_SERVICES. For more information about this setting, see the Cloud Foundry documentation.

Java clients can access environment information using `System.getenv("VCAP_SERVICES")`. You specify a JSON string as a value for VCAP_SERVICES. For applications bound to Apama, see the following sample JSON that contains a complete list of protocols:

```
{
  "VCAP_SERVICES": {
    "apama": [{
      "credentials": {
        "direct": "apama://10.245.0.137:15903",
        "http": "http://10.245.0.137:15903",
        "name": "42ae8035-feb5-480f-b0d5-d820fbb6bb54",
        "password": "SyuOL7,QWhyiGY;kIiL\u003e",
        "rest":
"http://10.245.0.137:8080/servers/42ae8035-feb5-480f-b0d5-d820fbb6bb54-0",
        "uriInfos": [{
          "host": "10.245.0.137",
          "port": 15903
        }],
        "userid": "cfu-14c878b4-613e-46b5-b6e9-abb7cdf6500c"
      },
      "label": "apama",
      "name": "apama",
      "plan": "gold",
      "provider": null,
      "syslog_drain_url": null,
      "tags": [
        "complex-event-processing",
        "cep",
        "event-processing",
        "streaming-analytics",
        "EPL"
      ]
    }]
  }
}
```

This sample lists all available protocols exposed in VCAP_SERVICES. The actual list is limited to the protocols supported by the service plan.

The supported protocols are:

Protocol or Key	Description
`direct`	Standard Apama protocol
`http`	Used by web clients

Protocol or Key	Description
`rest`	Used for injecting Apama projects

Also, available are lists of host or port information and uriInfos used by clients preferring their own URLs.

If the service is secured with a user ID and a password, this information is also provided in the `userid` and `password` fields.

Injecting the Apama Project

You can use one of the following methods to inject the Apama project into the Apama service:

- Inject by using the Apama buildpack. Use this method when you inject an Apama project for the first time. Software AG recommends using this method. See "Injecting the Apama Project by Using the Apama Buildpack" on page 61.
- Inject by using the CLI injection tool. Use this method to avoid redeploying a Cloud Foundry application that is already pushed. See "Injecting the Apama Project by Using the CLI Tool" on page 63.

Important: The inject script always restarts and clears the state of the Correlator, regardless of the method that you choose to inject the Apama project.

Injecting the Apama Project by Using the Apama Buildpack

To inject the Apama project into the Apama service, you can use an extension to the Cloud Foundry Java buildpack, named Apama buildpack. Use the following steps to perform the injection:

1. "Prepare the Apama Project for Injection" on page 61
2. "Add the Apama Project to the Deployed Application Package " on page 62
3. "Configure the Apama Buildpack" on page 62
4. "Deploy the Application that Uses the Apama Buildpack" on page 63

Important: You cannot combine the Apama buildpack with other Software AG buildpacks, for example the Universal Messaging buildpack.

Prepare the Apama Project for Injection

To prepare the Apama project for injection, make a "File System" export of the project from Software AG Designer and use the engine_deploy tool to create the necessary

YAML configurations for the project. The engine_deploy tool also copies project artifacts into the <apama-project-dir> directory. For example, use the following commands:

```
$ cd /opt/softwareag/Apama/bin/
$ . apama_env
$ ./engine_deploy <APAMA_PROJECT_DIR>/config/launch/MyApamaProject.deploy
--outputDeployDir <output-apama-project-dir>
```

Note: Use the <apama-project-dir> directory as a configuration parameter for the Apama buildpack.

Add the Apama Project to the Deployed Application Package

You must include the Apama project in the pushed application package for the injection to work. For example, use the following command to add the project directory, MyApamaProject, to the package my_app.war:

```
$ jar uvf my_app.war MyApamaProject
```

Configure the Apama Buildpack

The default configuration file for the Apama buildpack is located in /opt/softwareag/cloudfoundry/Apama/cf-Apama/buildpack/java/src/sag_injector.yml. If you want to change the default configuration, you can edit the following parameters:

Parameter	Description
`enabled`	When set to `true`, enables the Apama Injector. When set to `false`, disables the Apama Injector. Default: `true`
`services-bound`	Semicolon-separated list of Apama services. The services are bound to the cloud-native application that you push to Cloud Foundry. For example: apama; apama2
`apama-project-dir`	Directory in the application that you push to Cloud Foundry, on which the injection deployment is located. For example: MyApamaProject
`deploy-file`	Location of the deploy file, relative to `apama-project-dir`. For example: config/launch/MyApamaProject.deploy
`clear-state`	Clear the state of the Correlator on restart.

Parameter	Description
(deprecated)	Default: `true`

Use one of the following methods to pre-configure the Apama buildpack parameters:

- Configure the parameters on the cf CLI, for example:

  ```
  $ cf set-env <APP> JBP_CONFIG_SAG_INJECTOR '{ enabled: true,
  apama-project-dir: MyApamaProject,
  deploy-file: config/launch/MyApamaProject.deploy, services-bound:apama}'
  ```

- Configure the parameters in the manifest.yml file of the application, for example:

  ```
  ...
  env:
      JBP_CONFIG_SAG_INJECTOR :  '{ enabled: true,
  apama-project-dir: MyApamaProject,
  deploy-file: config/launch/MyApamaProject.deploy, services-bound:apama}'
  ```

Deploy the Application that Uses the Apama Buildpack

Use one of the following methods to instruct Cloud Foundry to use the Apama buildpack when pushing an application:

- In the cf CLI, add the following command line parameter:

  ```
  cf push <APP> -b sag-offline-java-buildpack
  ```

- In the manifest.yml file of the application, add the following setting:

  ```
  buildpack: sag-offline-java-buildpack
  ```

  For example:

  ```
  ---
  applications:
  - name: cf-um-test-app
    memory: 2G
    instances: 1
    host: my_app
    domain: bosh-lite.com
    path: my_app.war
    buildpack: sag-offline-java-buildpack
    stack: cflinuxfs2
    services:
    - apama
  env:
    JBP_CONFIG_SAG_INJECTOR: '{ enabled: true,
  apama-project-dir: MyApamaProject,
  deploy-file: config/launch/MyApamaProject.deploy, services-bound:apama}'
  ```

Injecting the Apama Project by Using the CLI Tool

You can use the CLI tool in the <install-directory>/cloudfoundry/Apama/cf-Apama/injector directory to inject the Apama project into the Apama service. Use the following steps to perform the injection:

1. "Prepare the Apama Project for Injection" on page 64

2. "Prepare the Parameters for Injection" on page 64
3. "Execute the Inject Script" on page 65

Important: You can inject the Apama project by using the CLI tool only if Software AG Enablement for Cloud Foundry is already installed on the machine.

Prepare the Apama Project for Injection

To prepare the Apama project for injection, make a "File System" export of the project from Software AG Designer and use the engine_deploy tool to create the necessary YAML configurations for the project. The engine_deploy tool also copies project artifacts into the <apama-project-dir> directory. For example, use the following commands:

```
$ cd /opt/softwareag/Apama/bin/
$ . apama_env
$ ./engine_deploy <APAMA_PROJECT_DIR>/config/launch/MyApamaProject.deploy
--outputDeployDir <output-apama-project-dir>
```

Prepare the Parameters for Injection

You can configure the following parameters of the inject script:

Parameter	Description
`-su | --service-username`	Username to log in to the Apama Service API. Take this value from VCAP_SERVICES.
`-sp | --service-password`	Password to log in to the Apama Service API. Take this value from VCAP_SERVICES.
`-f | --deploy-file`	Location of the Apama project deploy file. For example: `config/launch/ApamaProject.deploy`
`-n | --service-name`	Name of the Apama Service instance in which you want to inject the Apama project.
`-d | --deployment-directory`	Location of the Apama project.
`-u | --username`	Username to log in to Cloud Foundry API.
`-p | --password`	Password to log in to Cloud Foundry API.
`-a | --apamaURL`	The URL to connect to Apama. Take this value from VCAP_SERVICES.

Parameter	Description
	For example: `apama://10.246.0.8:15995`
`-c | --cf-api-url`	Cloud Foundry API URL. Default: `http://api.bosh-lite.com`
`-o | --client-id`	Cloud Foundry OAuth client AuthID. Default: `cf`
`--client-authid`	Cloud Foundry OAuth client ID. Default: `cf`
`--client-password`	Cloud Foundry OAuth client password.
`-h | --help`	Display the help message.
`--clear-state` (deprecated)	Clear the state of the Correlator on restart. Default: disabled

Execute the Inject Script

1. Go to <install-directory>/cloudfoundry/Apama/cf-Apama/injector.
2. Use the inject script to add the parameters of the injection.

The following code shows the command line input for injecting an Apama project.

```
/opt/softwareag/cloudfoundry/apama/cf-Apama/injector/inject
-su "cfu-33254cf1-ba89-4dfc-9457-b3cda2c65b73" -sp "xrp~d-0!C?;PE#Ct~!dg" -f
<ApamaProjectDir>/config/launch/<ProjectDeployFile>.deploy -n apama
-d <ApamaProjectDir>
-a http://10.246.0.8:8080/servers/bc5850b7-5398-4951-a3ef-31aba7b372a3-0
-c http://api.bosh-lite.com -u admin -p admin --clear-state
```

Collecting Diagnostic Information

Apama for Cloud Foundry provides a diagnostics tool that you can use to collect diagnostic information. The tool is located in the cloudfoundry/Apama/cf-Apama/scripts directory.

You can use the diagnostic tool to:

- Extend the logging for the Apama Broker and Service Instance Manager.

- Collect configuration information that Software AG Global Support requires to diagnose deployment issues.
- Generate logs from the Apama Broker and Service Instance Manager.

The diagnostic tool creates a Tar GZip file, named diagnosticLogs-<date>.tar.gz, in the cloudfoundry/Apama/cf-Apama/log directory.

The diagnostic tool has the following parameters:

Parameter	Description
`-c | --config`	Collect configuration-only files.
`-a | --all`	Collect both configuration and log files.
`-l | --log`	Collect log files from the Broker and Server nodes.
`-d | --debug <file>`	Upload a log4j.xml configuration file.

Extending Logging

You can extend the logging settings for the Cloud Foundry deployment by providing a log4j log configuration. The log4j configuration is deployed using the diagnostic utility and, by default, logs or audits each REST service call. You can find the following sample configuration files in the cloudfoundry/apama/cf-Apama/templates/samples/bosh-lite directory:

- log4j.properties. This is the minimal configuration that is deployed by default during deployment.
- log4j-debug.properties. This is a sample debug configuration.

Software AG recommends that you do not change the log settings unless you are requested to do so by Software AG Global Support.

Collection of Configuration Files

The configuration files contain information that helps you diagnose issues. The following files are part of the configuration collection:

- cloudfoundry/cf-Apama/cf-Apama/templates/cf-Apama-template.yml
- cloudfoundry/cf-Apama/cf-Apama/templates/custom/*.yml
- cloudfoundry/cf-Apama/cf-Apama/templates/custom/setup.properties
- cloudfoundry/cf-Apama/cf-Apama/deployment/*.yml

- cloudfoundry/cf-Apama/cf-Apama/scripts/setup.properties
- All files in the cloudfoundry/cf-Apama/cf-Apama/src/plans directory

Collection of Log Files

You can collect log files by using the `bosh logs` command in the bosh CLI. This command gets all log files from each Cloud Foundry BOSH virtual machine in the deployment.