



Course developed by
Pivotal Academy

Pivotal Cloud Foundry Developer v1.7

APPLICATION SECURITY GROUPS
AND LOG DRAIN

Application Security Groups

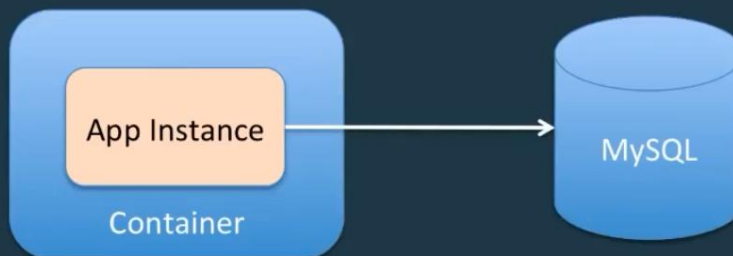
Setting up Application Security Groups is generally done by the admin, but as a developer you have tools to reason about the platform and how it is running your applications.

Agenda

1. Application Security Groups Overview

Application Security Groups

Are virtual firewalls that control egress/
outbound traffic for applications.

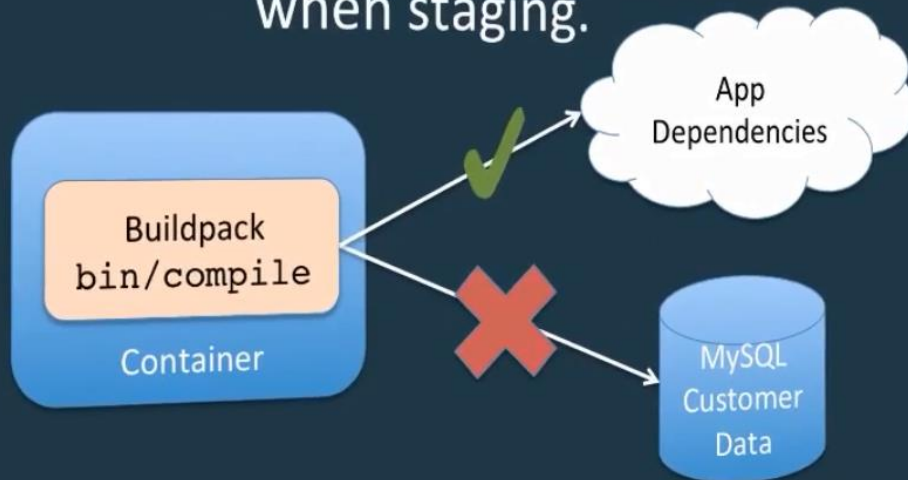


<https://docs.pivotal.io/pivotalcf/adminguide/app-sec-groups.html>



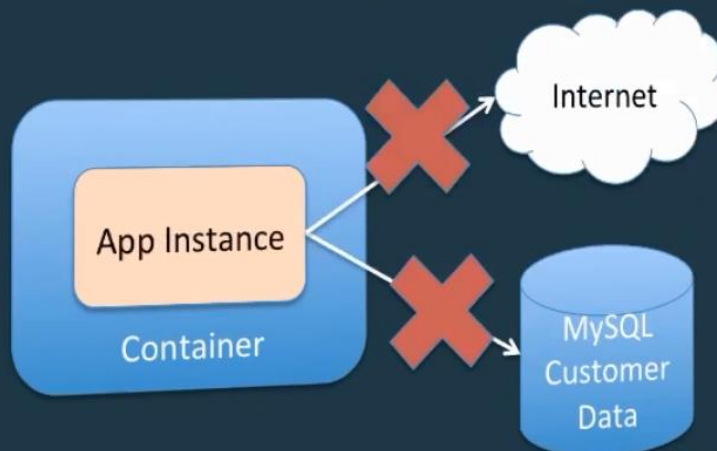
Staging Security Groups

Whitelist what the app should have access to when staging.



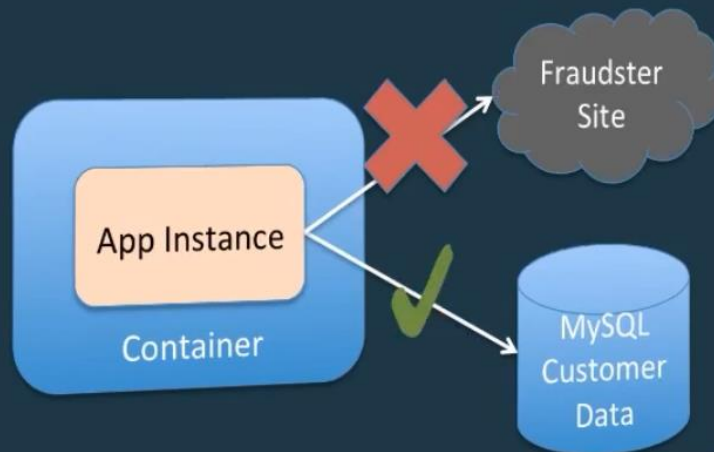
Running Security Groups

Whitelist what all apps should have access to when running.



Space Security Groups

Whitelist what the app should have access to when running.



Security Group Rules File

```
[
  {
    "protocol": "tcp",
    "destination": "10.0.11.0/24",
    "ports": "1-65535"
  },
  {
    "protocol": "udp",
    "destination": "10.0.11.0/24",
    "ports": "1-65535"
  }
]
```

You can then apply this to your CF environment

```
DROBERTS-MBPRO:attendee-service droberts$ cf t
API endpoint: https://api.run.haas-39.pez.pivotal.io (API version: 2.54.0)
User: droberts@pivotal.io
Org: dave
Space: dev
DROBERTS-MBPRO:attendee-service droberts$
```

We can see that we are currently logged in as a developer and not an admin.

```
DROBERTS-MBPRO:attendee-service droberts$ cf security-groups
Getting security groups as droberts@pivotal.io
OK

  Name      Organization  Space
#0  all_open

DROBERTS-MBPRO:attendee-service droberts$
```


We can use the **\$cf security-groups** command to see all the SGs currently running and are bound to our current Org and Space. We currently have 1 SG called **all_open**.

```
DROBERTS-MBPRO:attendee-service droberts$ cf security-groups
Getting security groups as droberts@pivotal.io
OK

  Name      Organization  Space
#0  all_open
DROBERTS-MBPRO:attendee-service droberts$ cf security-group all_open
Getting info for security group all_open as droberts@pivotal.io
OK

Name      all_open
Rules
[
  {
    "destination": "0.0.0.0-255.255.255.255",
    "protocol": "all"
  }
]

No spaces assigned
DROBERTS-MBPRO:attendee-service droberts$
```



We can investigate the details of an Application SG further using the **\$ cf security-group all_open** command as above, this shows that the SG opens all destinations and IP addresses and all protocols.

```
DROBERTS-MBPRO:attendee-service droberts$ cf login
API endpoint: https://api.run.haas-39.pez.pivotal.io

Email> admin

Password>
Authenticating...
OK

Select an org (or press enter to skip):
1. system
2. dave
3. sg-org

Org> 2
Targeted org dave

Targeted space dev

API endpoint: https://api.run.haas-39.pez.pivotal.io (API version: 2.54.0)
User: admin
Org: dave
Space: dev
DROBERTS-MBPRO:attendee-service droberts$
```

Let us now log in as an admin and again target the same Org and Space as earlier to see what checking the SGs look like.

```

DROBERTS-MBPRO:attendee-service droberts$ cf security-groups
Getting security groups as admin
OK

  Name      Organization  Space
#0  all_open
#1  metrics-api  system      metrics
DROBERTS-MBPRO:attendee-service droberts$

```

As an admin, using the **\$ cf security-groups** command, we now get a list of all the ASGs available within our PCF account.

```

DROBERTS-MBPRO:attendee-service droberts$ cf staging-security-groups
Acquiring staging security group as admin
OK

all_open
DROBERTS-MBPRO:attendee-service droberts$

```

The **\$ cf staging-security-groups** command will tell us all what SGs are active in the staging environment when we are creating the droplets. The `all_open` SG will allow us to be able to pull down all the dependencies that our application needs during staging.

```

DROBERTS-MBPRO:attendee-service droberts$ cf staging-security-groups
Acquiring staging security group as admin
OK

all_open
DROBERTS-MBPRO:attendee-service droberts$ cf running-security-groups
Acquiring running security groups as 'admin'
OK

all_open
DROBERTS-MBPRO:attendee-service droberts$

```

We can use the **\$ cf running-security-groups** command to see all the system-wide SGs as above. Both the staging-security-groups and the running-security-groups are system-wide SGs that apply across all Orgs and Spaces that can be targeted. A single SG can also be used within multiple Orgs and Spaces.

```

DROBERTS-MBPRO:attendee-service droberts$ cf unbind-running-security-group all_open
Unbinding security group all_open from defaults for running as admin
OK

TIP: Changes will not apply to existing running applications until they are restarted.
DROBERTS-MBPRO:attendee-service droberts$

```



We can lock down all egresses into our application by using the **\$ cf unbind-running-security-group all_open** command above. Next, let us see how this unbinding the `all_open` SG will affect our app.


```

DROBERTS-MBPRO:attendee-service droberts$ cf restart attendee-service
Stopping app attendee-service in org dave / space dev as admin...
OK

Starting app attendee-service in org dave / space dev as admin...

0 of 1 instances running, 1 starting
0 of 1 instances running, 1 starting
0 of 1 instances running, 1 starting
1 of 1 instances running

App started

OK

App attendee-service was started using this command `CALCULATED_MEMORY=$(PWD/.java-buildpack/open_jdk_jre/bin/java-
calculator-2.0.1_RELEASE -memorySizes=metaspace:64m.. -memoryWeights=heap:75,metaspace:10,native:10,stack:5 -memoryI
,metaspace:100% -totMemory=$MEMORY_LIMIT) && JAVA_OPTS="-Djava.io.tmpdir=$TMPDIR -XX:OnOutOfMemoryError=$PWD/.java-b
_jre/bin/killjava.sh $CALCULATED_MEMORY" && SERVER_PORT=$PORT eval exec $PWD/.java-buildpack/open_jdk_jre/bin/java $
D/./$PWD/.java-buildpack/spring_auto_reconfiguration/spring_auto_reconfiguration-1.10.0_RELEASE.jar org.springframework
arLauncher`

Showing health and status for app attendee-service in org dave / space dev as admin...

```

```

App started

OK

App attendee-service was started using this command `CALCULATED_MEMORY=$(PWD/.java-buildpack/open_jdk_jre/bin/java-
calculator-2.0.1_RELEASE -memorySizes=metaspace:64m.. -memoryWeights=heap:75,metaspace:10,native:10,stack:5 -memoryI
,metaspace:100% -totMemory=$MEMORY_LIMIT) && JAVA_OPTS="-Djava.io.tmpdir=$TMPDIR -XX:OnOutOfMemoryError=$PWD/.java-b
_jre/bin/killjava.sh $CALCULATED_MEMORY" && SERVER_PORT=$PORT eval exec $PWD/.java-buildpack/open_jdk_jre/bin/java $
D/./$PWD/.java-buildpack/spring_auto_reconfiguration/spring_auto_reconfiguration-1.10.0_RELEASE.jar org.springframework
arLauncher`

Showing health and status for app attendee-service in org dave / space dev as admin...
OK

requested state: started
instances: 1/1
usage: 512M x 1 instances
urls: attendee-service-monochromatic-guarantee.cfapps.haas-39.pcz.pivotal.io
last uploaded: Fri May 27 21:36:43 UTC 2016
stack: unknown
buildpack: java-buildpack=v3.6-offline-https://github.com/cloudfoundry/java-buildpack
1 open-jdk-like-memory-calculator=2.0.1_RELEASE spring-auto-reconfiguration=1.10.0_RELEASE

#0 state since cpu memory disk disk usage
running 10-05-31 11:19:58 AM 0.1% 449.5M of 512M 152.4M of 1G 0%

DROBERTS-MBPRO:attendee-service droberts$

```

```

DROBERTS-MBPRO:attendee-service droberts$ cf logs attendee-service
Connected, tailing logs for app attendee-service in org dave / space dev as admin...

```

We then start tailing the logs coming out of the attendee-service using the command **\$ cf logs attendee-service**

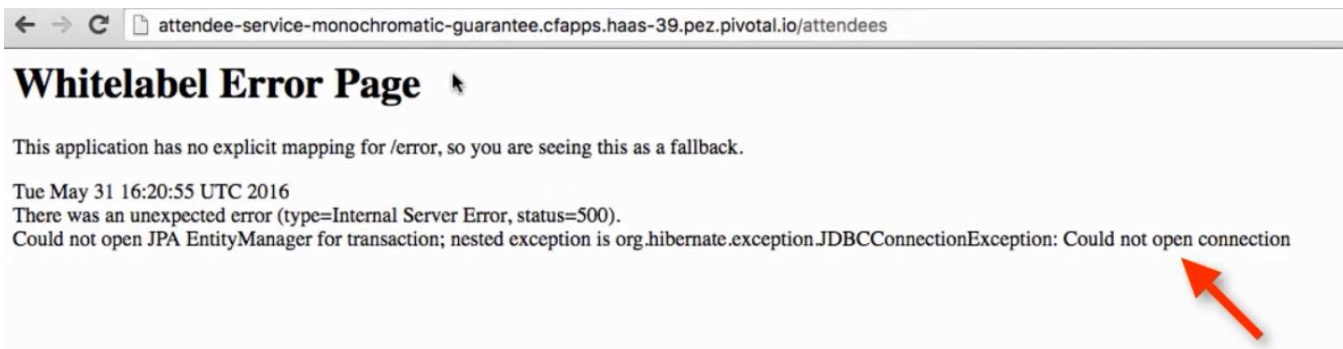
```

DROBERTS-MBPRO:attendee-service droberts$ cf aosp
'apsp' is not a registered command. See 'cf help'
DROBERTS-MBPRO:attendee-service droberts$ cf apps
Getting apps in org dave / space dev as admin...
OK

name requested state instances memory disk urls
articulate started 1/1 512M 1G articulate-turbosupercharged-spinneret.cfapps.haas-39.pcz.pivotal.io
attendee-service started 1/1 512M 1G attendee-service-monochromatic-guarantee.cfapps.haas-39.pcz.pivotal.io

DROBERTS-MBPRO:attendee-service droberts$

```



We then call one of the REST APIs on the attendee-service. We can see that we get a JDBC connection error because we cannot open a connection because the SG was unbound and there is no outbound connection for our application.

```
2016-05-31T11:20:54.72-0500 [APP/0] OUT Note: further occurrences of Cookie errors will be logged at DEBUG level
2016-05-31T11:20:54.73-0500 [APP/0] OUT 2016-05-31 16:20:54.738 INFO 24 --- [nio-8080-exec-3] o.a.c.c.C.[Tomcat
] : Initializing Spring FrameworkServlet 'dispatcherServlet'
2016-05-31T11:20:54.73-0500 [APP/0] OUT 2016-05-31 16:20:54.738 INFO 24 --- [nio-8080-exec-3] o.s.web.servlet.
: FrameworkServlet 'dispatcherServlet': initialization started
2016-05-31T11:20:54.76-0500 [APP/0] OUT 2016-05-31 16:20:54.762 INFO 24 --- [nio-8080-exec-3] o.s.web.servlet.
: FrameworkServlet 'dispatcherServlet': initialization completed in 24 ms
2016-05-31T11:20:55.87-0500 [APP/0] OUT 2016-05-31 16:20:55.879 WARN 24 --- [nio-8080-exec-3] o.h.engine.jdbc.
elper : SQL Error: 0, SQLState: 08S01
2016-05-31T11:20:55.88-0500 [APP/0] OUT 2016-05-31 16:20:55.879 ERROR 24 --- [nio-8080-exec-3] o.h.engine.jdbc.
elper : Communications link failure
2016-05-31T11:20:55.88-0500 [APP/0] OUT The last packet sent successfully to the server was 0 milliseconds ago.
ot received any packets from the server.
2016-05-31T11:20:55.88-0500 [APP/0] OUT 2016-05-31 16:20:55.889 ERROR 24 --- [nio-8080-exec-3] o.a.c.c.C.[.].[/
let] : Servlet.service() for servlet [dispatcherServlet] in context with path [] threw exception [Request process
d exception is org.springframework.transaction.CannotCreateTransactionException: Could not open JPA EntityManager fo
sted exception is org.hibernate.exception.JDBCConnectionException: Could not open connection] with root cause
2016-05-31T11:20:55.88-0500 [APP/0] OUT java.net.ConnectException: Connection refused
2016-05-31T11:20:55.88-0500 [APP/0] OUT at java.net.PlainSocketImpl.socketConnect(Native Method) ~[na:1.8.0_
2016-05-31T11:20:55.88-0500 [APP/0] OUT at java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl
1.8.0_71-)
2016-05-31T11:20:55.88-0500 [APP/0] OUT at java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl
) ~[na:1.8.0_71-)
2016-05-31T11:20:55.88-0500 [APP/0] OUT at java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl
8.0_71-)
2016-05-31T11:20:55.88-0500 [APP/0] OUT at java.net.SocksSocketImpl.connect(SocksSocketImpl:392) ~[na:1
2016-05-31T11:20:55.88-0500 [APP/0] OUT at java.net.Socket.connect(Socket:392) ~[na:1
2016-05-31T11:20:55.88-0500 [APP/0] OUT at com.mysql.jdbc.StandardSocketFactory.connect(StandardSocketFactory
ql-connector-java-5.1.38.jar!/:5.1.38]
2016-05-31T11:20:55.88-0500 [APP/0] OUT at com.mysql.jdbc.MysqlIO.<init>(MysqlIO:392) ~[na:1
/:5.1.38]
```


All the egress traffic into our app has been locked down because we have unbound the **all_open** ASG. We can instead create a space SG that will allow us add some egress traffic to our application for connecting to MySQL.

```

DROBERTS-MBPRO:attendee-service droberts$ cf env attendee-service
Getting env variables for app attendee-service in org dave / space dev as admin...
OK

System-Provided:
{
  "VCAP_SERVICES": {
    "p-mysql": [
      {
        "credentials": {
          "hostname": "10.65.188.80",
          "jdbcUrl": "jdbc:mysql://10.65.188.80:3306/cf_9b4bba91_c2cc_4b82_a9ce_61506b60dce9?user=4otnLyRKEv0bz8r3\u0026password=1Ff1XBQGYxqLUjcn",
          "name": "cf_9b4bba91_c2cc_4b82_a9ce_61506b60dce9",
          "password": "1Ff1XBQGYxqLUjcn",
          "port": 3306,
          "uri": "mysql://4otnLyRKEv0bz8r3:1Ff1XBQGYxqLUjcn@10.65.188.80:3306/cf_9b4bba91_c2cc_4b82_a9ce_61506b60dce9?reconnect=true",
          "username": "4otnLyRKEv0bz8r3"
        },
        "label": "p-mysql",
        "name": "attendee-mysql",
        "plan": "100mb-dev",
        "provider": null,
        "syslog_drain_url": null,
        "tags": [
          "mysql",
          "relational"
        ]
      }
    ]
  }
}

```



We can look at the application's environment using the `$ cf env attendee-service` command to determine the ports that need to be opened. We can see the application IP address **10.65.188.80** and the MySQL port **3306** that should allow egress traffic, we can then create an ASG file called **asg.json** with this 2 information on our local file system.

```

DROBERTS-MBPRO:attendee-service droberts$ nano asg.json
DROBERTS-MBPRO:attendee-service droberts$

```

```

GNU nano 2.0.6                                File: asg.json
[
  {
    "destination": "10.65.188.80",
    "ports": "3306",
    "protocol": "tcp"
  }
]

```

```

DROBERTS-MBPRO:attendee-service droberts$ nano asg.json
DROBERTS-MBPRO:attendee-service droberts$ ls
asg.json                                attendee-service-0.0.1-SNAPSHOT.jar
DROBERTS-MBPRO:attendee-service droberts$

```

We can now apply this new ASG within our PCF environment as below

```

DROBERTS-MBPRO:attendee-service droberts$ cf create-security-group mysql-dnr ./asg.json
Creating security group mysql-dnr as admin
OK
DROBERTS-MBPRO:attendee-service droberts$

```

We use the `$ cf create-security-group mysql-dnr ./asg.json` command to create a new ASG called **mysql-dnr** from our **asg.json** file in the path specified as above.


```

DROBERTS-MBPRO:attendee-service droberts$ cf create-security-group mysql-dnr ./asg.json
Creating security group mysql-dnr as admin
OK
DROBERTS-MBPRO:attendee-service droberts$ cf bind-security-group mysql-dnr dave dev
Assigning security group mysql-dnr to space dev in org dave as admin...
OK

TIP: Changes will not apply to existing running applications until they are restarted.
DROBERTS-MBPRO:attendee-service droberts$

```

We now use the **\$ cf bind-security-group mysql-dnr dave dev** command to bind the ASG to the **dave** Org and the **dev** Space.

```

DROBERTS-MBPRO:attendee-service droberts$ cf restart attendee-service
Stopping app attendee-service in org dave / space dev as admin...
OK

Starting app attendee-service in org dave / space dev as admin...

0 of 1 instances running, 1 starting
0 of 1 instances running, 1 starting
1 of 1 instances running

App started

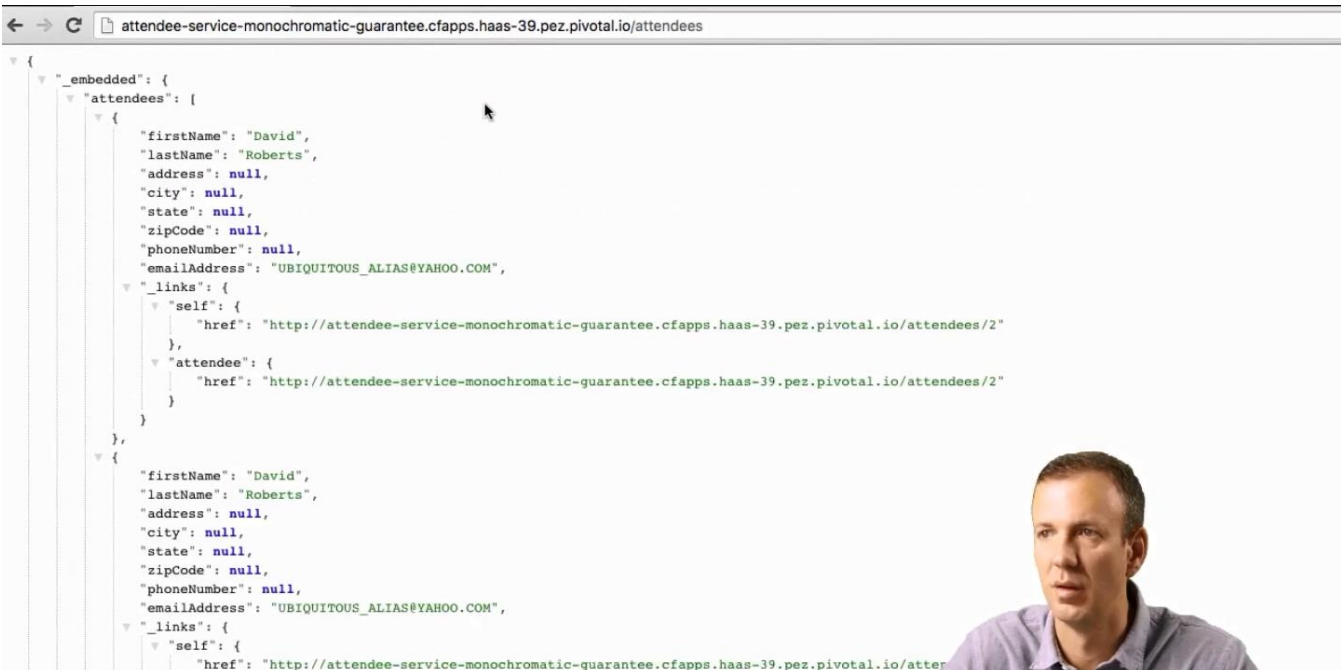
OK

App attendee-service was started using this command `CALCULATED_MEMORY=$(PWD/.java-buildpack/open_jdk_jre/bin/java-calculator-2.0.1_RELEASE -memorySizes=metaspace:64m.. -memoryWeights=heap:75,metaspace:10,native:10,stack:5 -memoryLimits=metaspace:100% -totMemory=$MEMORY_LIMIT) && JAVA_OPTS="-Djava.io.tmpdir=$TMPDIR -XX:OnOutOfMemoryError=$PWD/.java-buildpack/open_jdk_jre/bin/killjava.sh $CALCULATED_MEMORY" && SERVER_PORT=$PORT eval exec $PWD/.java-buildpack/open_jdk_jre/bin/java $JAVA_OPTS -Dorg.springframework.boot.autoconfigure.SpringBootApplicationLauncher`

Showing health and status for app attendee-service in org dave / space dev as admin...

```

We then restart the application using the **\$ cf restart attendee-service** command to have all the new security group rules take effect.



```

{
  "_embedded": {
    "attendees": [
      {
        "firstName": "David",
        "lastName": "Roberts",
        "address": null,
        "city": null,
        "state": null,
        "zipCode": null,
        "phoneNumber": null,
        "emailAddress": "UBIQUITOUS_ALIAS@YAHOO.COM",
        "_links": {
          "self": {
            "href": "http://attendee-service-monochromatic-guarantee.cfapps.haas-39.pez.pivotal.io/attendees/2"
          },
          "attendee": {
            "href": "http://attendee-service-monochromatic-guarantee.cfapps.haas-39.pez.pivotal.io/attendees/2"
          }
        }
      },
      {
        "firstName": "David",
        "lastName": "Roberts",
        "address": null,
        "city": null,
        "state": null,
        "zipCode": null,
        "phoneNumber": null,
        "emailAddress": "UBIQUITOUS_ALIAS@YAHOO.COM",
        "_links": {
          "self": {
            "href": "http://attendee-service-monochromatic-guarantee.cfapps.haas-39.pez.pivotal.io/attendees/2"
          },
          "attendee": {
            "href": "http://attendee-service-monochromatic-guarantee.cfapps.haas-39.pez.pivotal.io/attendees/2"
          }
        }
      }
    ]
  }
}

```

We can now see that our attendee-service is now allowed to make an outbound connection to MySQL for data over the allowed port and IP.

```
DROBERTS-MBPRO:attendee-service droberts$ cf login
API endpoint: https://api.run.haas-39.pez.pivotal.io

Email> droberts@pivotal.io

Password>
Authenticating...
Credentials were rejected, please try again.

Password>
Authenticating...
OK

Targeted org dave

Targeted space dev

API endpoint: https://api.run.haas-39.pez.pivotal.io (API version: 2.54.0)
User: droberts@pivotal.io
Org: dave
Space: dev
DROBERTS-MBPRO:attendee-service droberts$
```

We can now login as a developer to see the same SG effect

```
DROBERTS-MBPRO:attendee-service droberts$ cf security-groups
Getting security groups as droberts@pivotal.io
OK

#0  Name      Organization  Space
mysql-dnr  dave         dev
DROBERTS-MBPRO:attendee-service droberts$
```

We can see the newly applied SG

```
DROBERTS-MBPRO:attendee-service droberts$ cf security-groups
Getting security groups as droberts@pivotal.io
OK

#0  Name      Organization  Space
mysql-dnr  dave         dev
DROBERTS-MBPRO:attendee-service droberts$ cf security-group mysql-dnr
Getting info for security group mysql-dnr as droberts@pivotal.io
OK

Name      mysql-dnr
Rules
[
  {
    "destination": "10.65.188.80",
    "ports": "3306",
    "protocol": "tcp"
  }
]

#0  Organization  Space
dave  dev
DROBERTS-MBPRO:attendee-service droberts$
```

The new SG reflects correctly what we defined in our JSON file at creation.

Application Security Groups

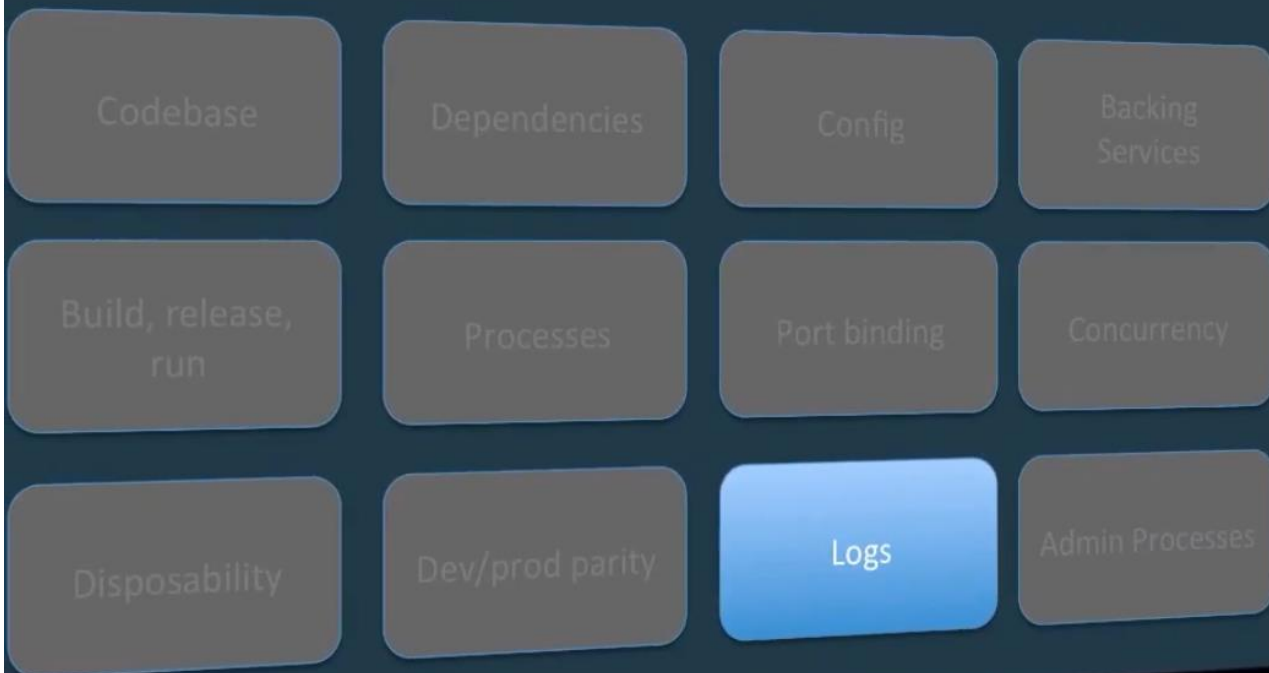
Recap

Log Drain

Agenda

1. Cloud Native Apps
2. Loggregator Review

The 12 Factors



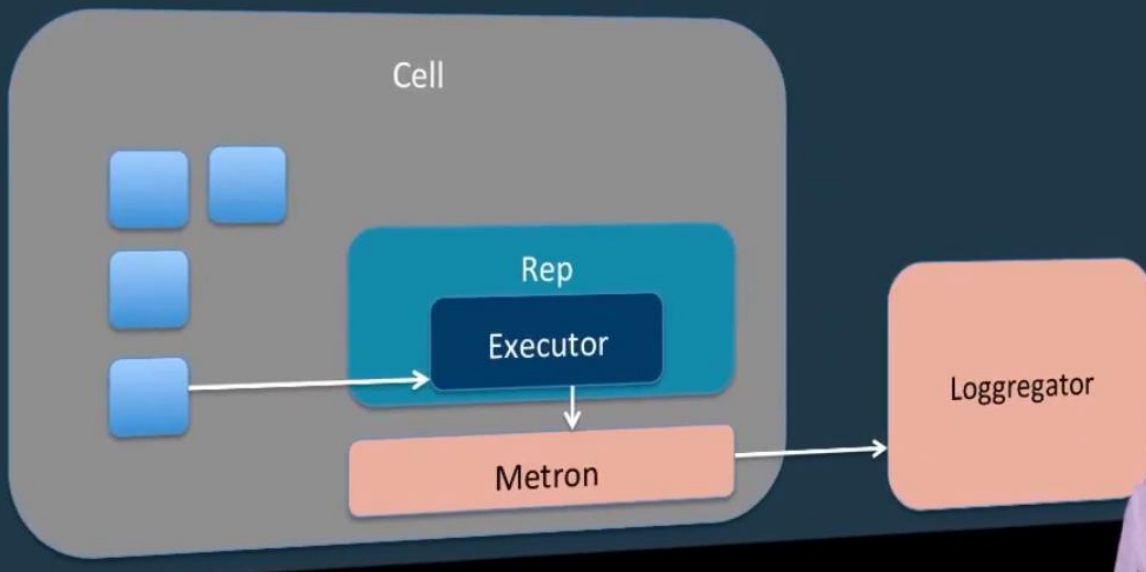
Logs

Treat logs as event streams.

This allows us to easily get our application logs into another system that will allow us investigate the data further, all your app needs to do is to write out the logs to stdout.

Loggregator: Metron

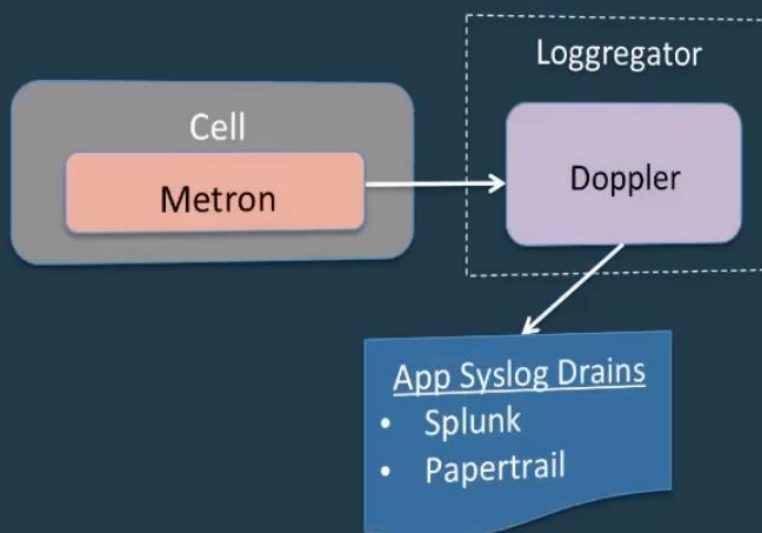
Forwards logs to the Loggregator subsystem.



The Executor captures all the stdouts and stderrs from your application streams and sends them to Metron that forwards them to Loggregator.

Loggregator: Doppler

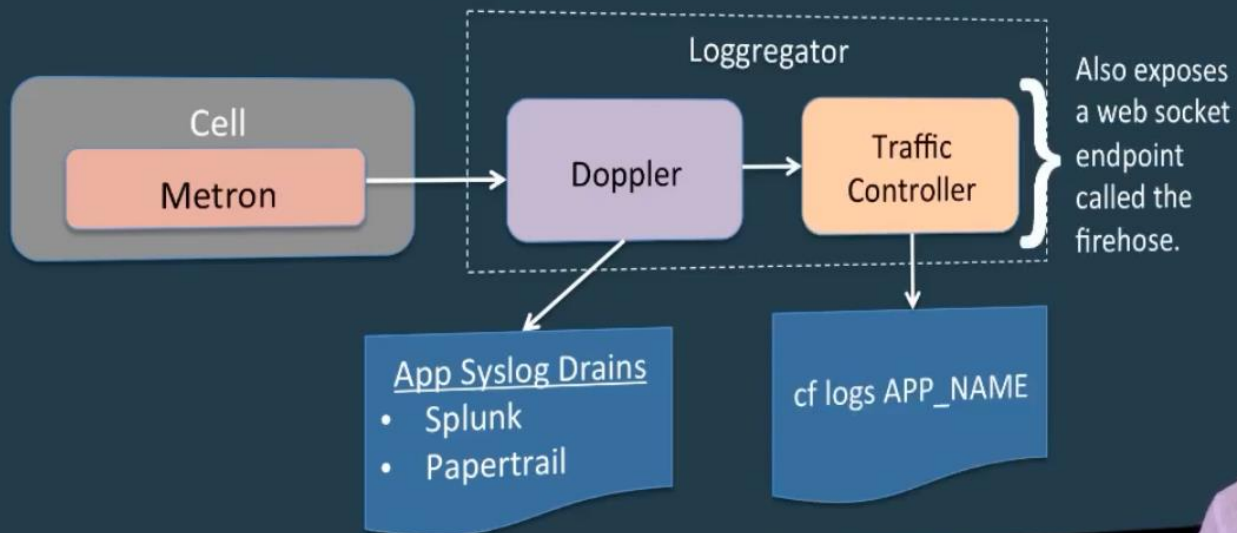
Gathers logs from Metron.



Doppler is responsible for forwarding the logs to the different log drains that you define for your applications.

Loggregator: Traffic Controller

Handles client requests for logs.

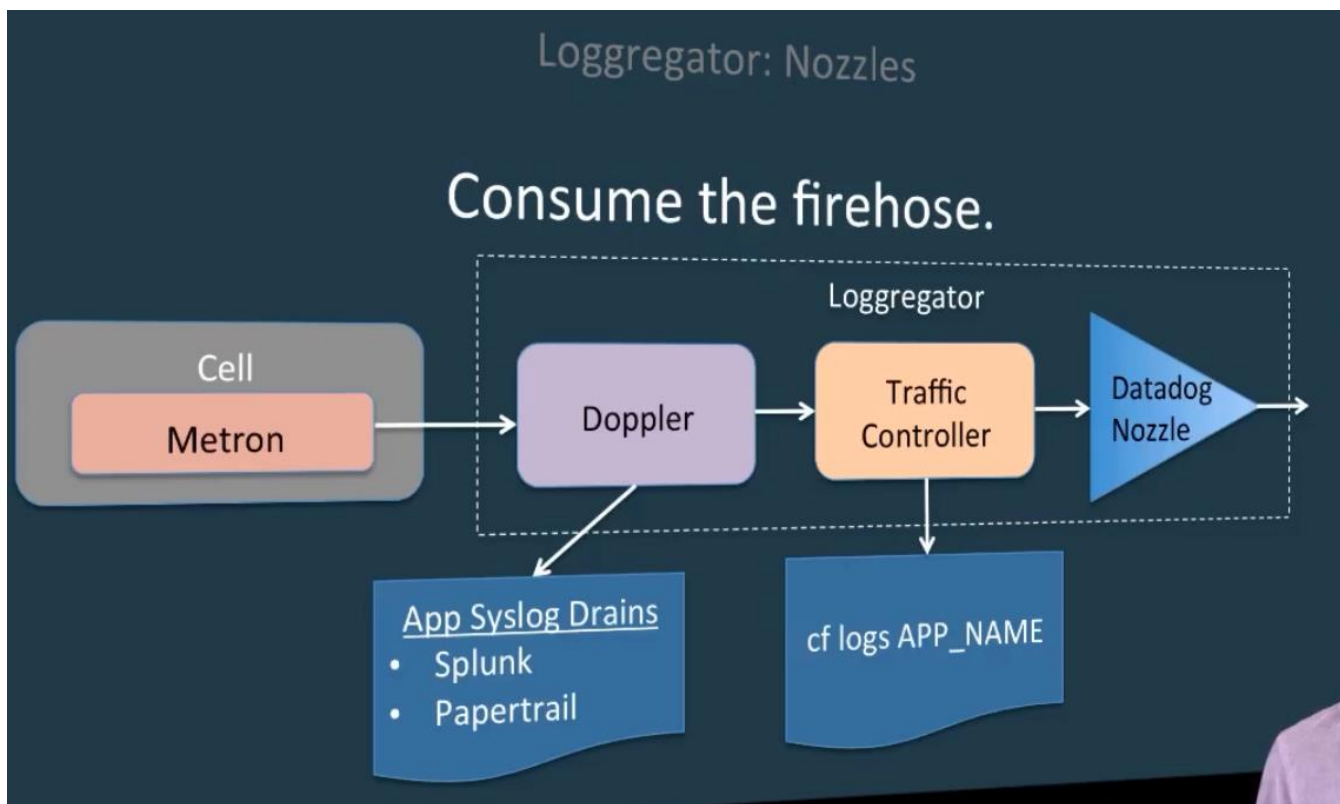


The Traffic Controller is the component that you are interacting with when you run the **\$ cf logs** command from the CF CLI.

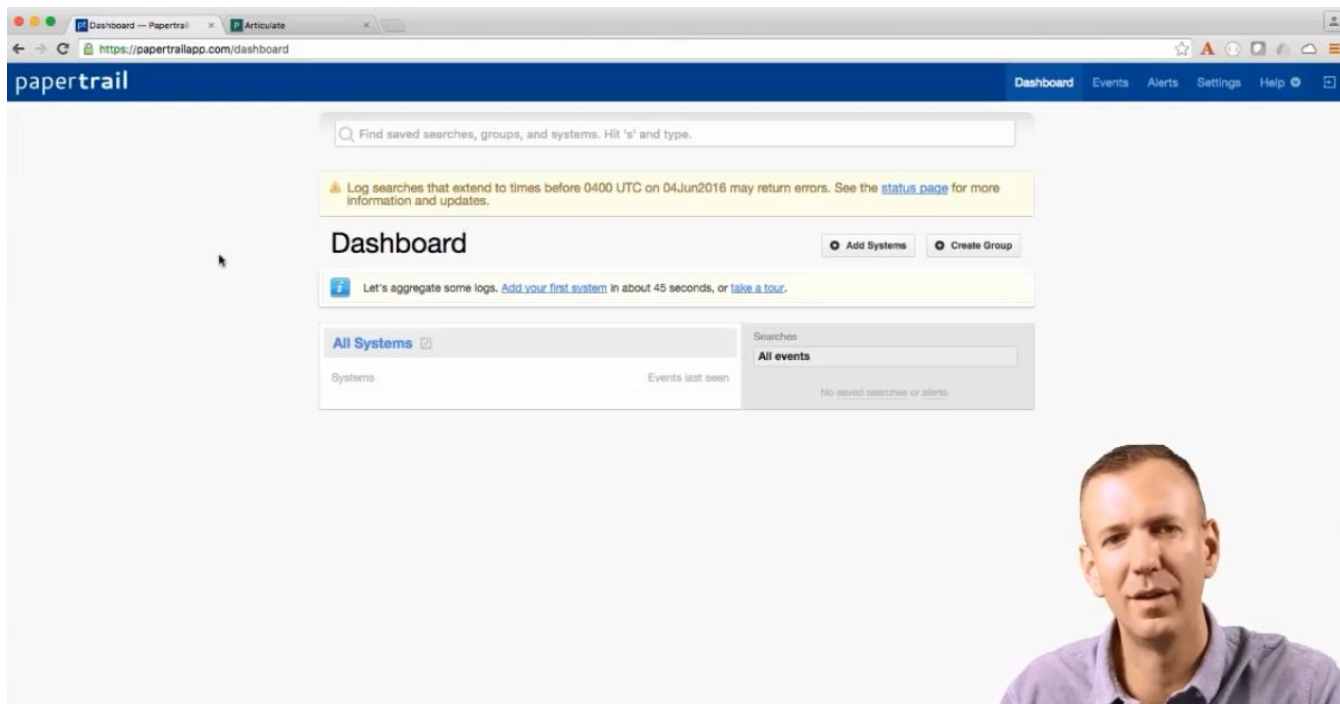
Loggregator: Firehose

A websocket endpoint that exposes app logs, container metrics and ER component metrics.

Does not include ER component logs.



The firehose gets consumed by components called Nozzles. Next, let us see a quick demonstration of a log drain.



We have set up an account with PaperTrail, a log management 3rd party service. We will then set up our app to send logs to PaperTrail.

Add System

Usually, no explicit configuration is required on Papertrail - just start sending logs. When Papertrail receives a message from a hostname that is not already present in your account, the system will be automatically added (see [Setup Systems](#)).

In a few cases, one hostname does not represent one system or service. Here's a few other ways that Papertrail can identify messages.

❗ One of these methods should work for all cases. If they don't, [we want to know](#).

Choose your situation:

☐ **A My syslogd only uses the default port**

GNU syslogd and some embedded devices will only log to port 514. A few old Linux distro versions use GNU syslogd (mostly CentOS and Gentoo).

☒ **B I use Cloud Foundry**

Register each app separately. Use Heroku? [Here's how](#).

☐ **C My system's hostname changes**

In rare cases, one system may change hostnames frequently. For example, a roaming laptop which sets its hostname based on DHCP (and roams across networks).

We'll provide an app-specific syslog drain and step-by-step setup for [Cloud Foundry](#).

Let's create a destination for this app.

What should we call it?

log-drain


Alphanumeric. Does not need to match app name.

Save

Setup log-drain...

Edit Settings

✓ System created.

log-drain will log to logs4.papertrailapp.com:14405. 

I'm using...

Unix & Linux

Text files, Apache, MySQL, Docker & more

BSD & OS X

Windows

🔍 Not shown here?

1 See which logger your system uses. Run:

```
ls -ld /etc/*syslog*
```

Which filename is listed?

✓ **rsyslog.conf**

[syslog-ng.conf](#)

[syslog.conf](#)

[Other or no file](#)

Setup: rsyslog.conf

2

As root, edit /etc/rsyslog.conf with a text editor (like pico or vi). Paste this line at the end:

We now have URL to send our logs to from the articulate application.

```
DROBERTS-MBPRO:articulate droberts$ cf create-user-provided-service articulate-log-drain -l syslog://logs4.papertrailapp.com:14405
Creating user provided service articulate-log-drain in org dave / space dev as droberts@pivotal.io...
OK
DROBERTS-MBPRO:articulate droberts$
```

We create our **articulate-log-drain** log-drain service for the articulate application using the command **\$ cf create-user-provided-service articulate-log-drain -l syslog://<our papertrail URL logs endpoint>** above.

```
DROBERTS-MBPRO:articulate droberts$ cf bind-service articulate articulate-log-drain
Binding service articulate-log-drain to app articulate in org dave / space dev as droberts@pivotal.io...
OK
TIP: Use 'cf restage articulate' to ensure your env variable changes take effect
DROBERTS-MBPRO:articulate droberts$
```

We then bind our new **articulate-log-drain** service to our **articulate** application using the **\$ cf bind-service articulate articulate-log-drain** command above. Our points are now going to start flowing into our PaperTrail endpoint for further analysis.

The left screenshot shows the PaperTrail 'Setup log-drain...' page. It includes a 'System created' status, a log-drain URL, and instructions for configuring rsyslog. The right screenshot shows the 'Welcome to Articulate!' page, which includes an application architecture diagram and a video player.

Setup log-drain...

log-drain will log to logs4.papertrailapp.com:14405.

I'm using...

Unix & Linux Text files, Apache, MySQL, Docker & more BSD & OS X Windows Q Not shown here?

1 See which logger your system uses. Run:

```
ls -d /etc/*syslog*
```

Which filename is listed?

rsyslog.conf syslog-ng.conf syslog.conf Other or no file

Setup: rsyslog.conf

2 As root, edit /etc/rsyslog.conf with a text editor (like pico or vi). Paste this line at the end:

```
*.* @logs4.papertrailapp.com:14405
```

3 Restart rsyslog to re-read the config file:

```
sudo service rsyslog restart
```

Optionally, use [TLS encryption](#).

Welcome to Articulate!

The purpose of this application is to articulate some basic concepts and capabilities of the Pivotal Cloud Foundry platform, specifically the Elastic Runtime which is responsible for running application workloads.

Application Architecture

articulate is a web application that exposes friendly, browsable user interface. However, it does not work with data directly. It depends on the attendee-service application to manage data. The attendee-service persists data to a MySQL database.

How to use this Application

Each menu item above links to a page that helps demonstrate a set of capabilities provided by the platform. The last item, Spring Boot, highlights capabilities that come with Spring Boot to help build production ready microservices in minutes.

Each page has the same layout with the Accordion control and...

1. **Application Environment Information** - This provides information about the application environment when running inside PCF. You can see the Application Name, Instance Index, Container Address, Cell Address, and Java Version. This is useful to show things like load balancing, self healing, service brokering, etc.

2. **Description** - additional context for the given page.

3. **The Twelve-Factor App** - a methodology for building modern applications. Links to applicable factors will be provided.

Application Environment Information

Application Name: articulate
Instance Index: 0
Container Address: 10.254.0.6:8080
Cell Address: 10.65.168.46:6026
Java Version: 1.8.0_71

We can then generate some traffic using the articulate application

The left screenshot shows the PaperTrail 'Dashboard' page. It includes a search bar, a status message about log searches, and a list of systems. The right screenshot shows the 'Welcome to Articulate!' page, which is identical to the one in the previous block.

Dashboard

Find saved searches, groups, and systems. Hit 's' and type.

Log searches that extend to times before 0400 UTC on 04Jun2016 may return errors. See the [status page](#) for more information and updates.

Have more systems or log files? [Add more systems](#) or aggregate [text log files](#).

All Systems ☒ Searches All events

Systems Events last seen No saved searches or alerts

log-drain

Welcome to Articulate!

The purpose of this application is to articulate some basic concepts and capabilities of the Pivotal Cloud Foundry platform, specifically the Elastic Runtime which is responsible for running application workloads.

Application Architecture

articulate is a web application that exposes friendly, browsable user interface. However, it does not work with data directly. It depends on the attendee-service application to manage data. The attendee-service persists data to a MySQL database.

How to use this Application

Each menu item above links to a page that helps demonstrate a set of capabilities provided by the platform. The last item, Spring Boot, highlights capabilities that come with Spring Boot to help build production ready microservices in minutes.

Each page has the same layout with the Accordion control and...

1. **Application Environment Information** - This provides information about the application environment when running inside PCF. You can see the Application Name, Instance Index, Container Address, Cell Address, and Java Version. This is useful to show things like load balancing, self healing, service brokering, etc.

2. **Description** - additional context for the given page.

3. **The Twelve-Factor App** - a methodology for building modern applications. Links to applicable factors will be provided.

Application Environment Information

Application Name: articulate
Instance Index: 0
Container Address: 10.254.0.6:8080
Cell Address: 10.65.168.46:6026
Java Version: 1.8.0_71

papertrail

Dashboard Events Alerts Settings Help

Example: "access denied" 1.2.3.4 - sshd

Search log-drain

Articulate Scale & HA Services Blue-Green Spring Boot

Welcome to Articulate!

The purpose of this application is to articulate some basic concepts and capabilities of the Pivotal Cloud Foundry platform, specifically the Elastic Runtime which is responsible for running application workloads.

Application Architecture

articulate is a web application that exposes friendly, browsable user interface. However, it does not work with data directly. It depends on the `attendee-service` application to manage data. The `attendee-service` persists data to a MySQL database.

How to use this Application

Each menu item above links to a page that helps demonstrate a set of capabilities provided by the platform. The last item, Spring Boot, highlights capabilities that come with Spring Boot to help build production ready microservices in minutes.

Each page has the same layout with the Accordion control and:

1. Application Environment Information - This provides information about the application environment when running inside PCF. You can see the Application Name, Instance Index, Container Address, Cell Address, and Java Version.
2. Description - additional context for the given page.
3. The Twelve-Factor App - a methodology for building microservices. Links to applicable factors will be provided.

Application Environment Information

Application Name: articulate
Instance Index: 0
Container Address: 10.254.0.6:8080
Cell Address: 10.65.188.46:60262
Java Version: 1.8.0_71

AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36 10.65.188.32:42437 x_forwarded_for: "66.68.129.103, 10.65.188.32" x_forwarded_proto: "http" vcap_request_id: ae0a9f93-2ba8-442c-4c5c-e4e3549f9125 response_time: 0.00438704 app_id: 91f1db69-32d1-46c4-a5af-5f69f9bcb74f

Jun 08 07:11:30 log-drain 91f1db69-32d1-46c4-a5af-5f69f9bcb74f [RTR]: articulate-turbosupercharged-spinneret.cfapps.haas-39.pez.pivotal.io - [08/06/2016:14:11:30.416 +0000] "GET /tutorial/index-desc.html HTTP/1.1" 200 0 906 "http://articulate-turbosupercharged-spinneret.cfapps.haas-39.pez.pivotal.io/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36" 10.65.188.32:42431 x_forwarded_for: "66.68.129.103, 10.65.188.32" x_forwarded_proto: "http" vcap_request_id: ce6f9a4f-f5c9-464b-5c08-be1a927a088b response_time: 0.003095673 app_id: 91f1db69-32d1-46c4-a5af-5f69f9bcb74f

Jun 08 07:11:30 log-drain 91f1db69-32d1-46c4-a5af-5f69f9bcb74f [RTR]: articulate-turbosupercharged-spinneret.cfapps.haas-39.pez.pivotal.io - [08/06/2016:14:11:30.418 +0000] "GET /tutorial/index-12f.html HTTP/1.1" 200 0 71 "http://articulate-turbosupercharged-spinneret.cfapps.haas-39.pez.pivotal.io/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36" 10.65.188.32:42429 x_forwarded_for: "66.68.129.103, 10.65.188.32" x_forwarded_proto: "http" vcap_request_id: 77709bbd-6d14-45bf-583c-4aa550b5c19c response_time: 0.003121476 app_id: 91f1db69-32d1-46c4-a5af-5f69f9bcb74f

Jun 08 07:11:30 log-drain 91f1db69-32d1-46c4-a5af-5f69f9bcb74f [RTR]: articulate-turbosupercharged-spinneret.cfapps.haas-39.pez.pivotal.io - [08/06/2016:14:11:30.505 +0000] "GET /images/P_Mark_WhiteOnTeal.png HTTP/1.1" 200 0 3901 "http://articulate-turbosupercharged-spinneret.cfapps.haas-39.pez.pivotal.io/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36" 10.65.188.32:42431 x_forwarded_for: "66.68.129.103, 10.65.188.32" x_forwarded_proto: "http" vcap_request_id: 7b60f92a-4136-489d-5b44-6ab2466badff response_time: 0.00439153 app_id: 91f1db69-32d1-46c4-a5af-5f69f9bcb74f

Jun 08 07:11:30 log-drain 91f1db69-32d1-46c4-a5af-5f69f9bcb74f [RTR]: articulate-turbosupercharged-spinneret.cfapps.haas-39.pez.pivotal.io - [08/06/2016:14:11:30.504 +0000] "GET /images/index1.png HTTP/1.1" 200 0 173662 "http://articulate-turbosupercharged-spinneret.cfapps.haas-39.pez.pivotal.io/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36" 10.65.188.32:42429 x_forwarded_for: "66.68.129.103, 10.65.188.32" x_forwarded_proto: "http" vcap_request_id: 2263d64b-dfc6-4e1b-879291d4 response_time: 0.00721744 app_id: 91f1db69-32d1-46c4-a5af-5f69f9bcb74f

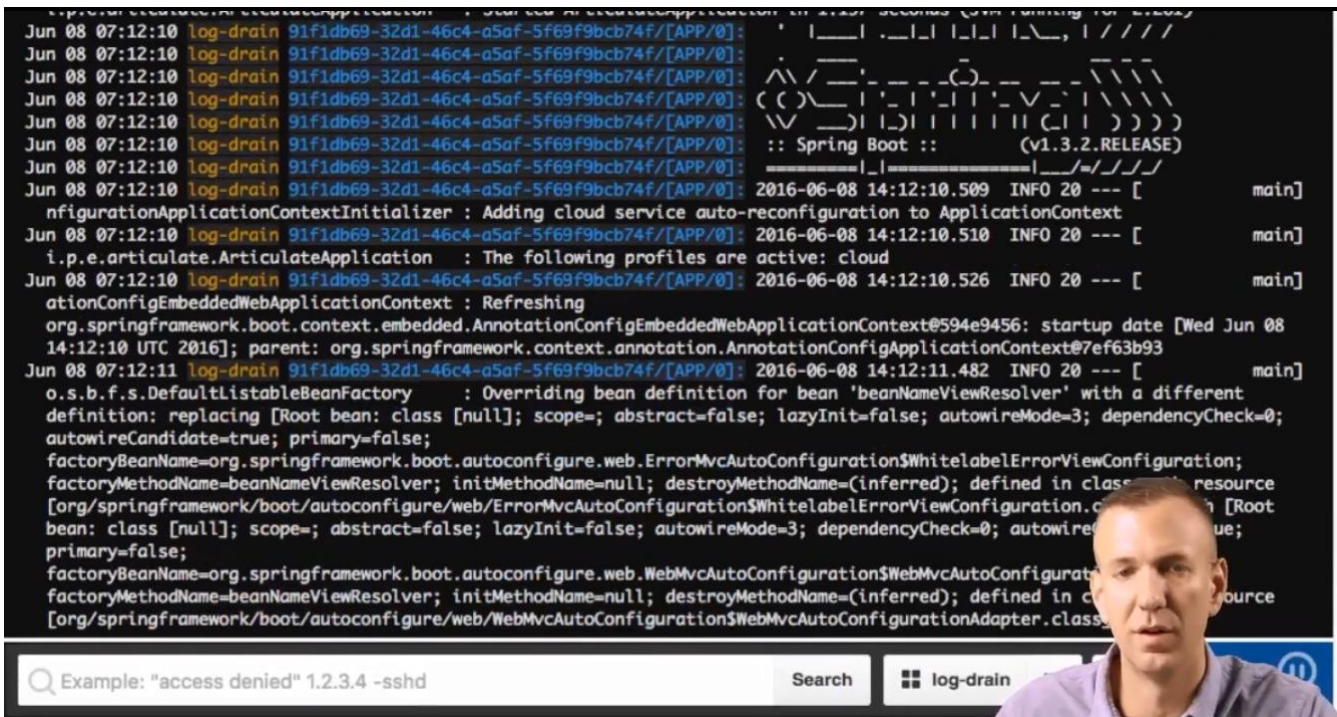
Example: "access denied" 1.2.3.4 - sshd Search log-drain

We can now start seeing our logs drained from the articulate application.

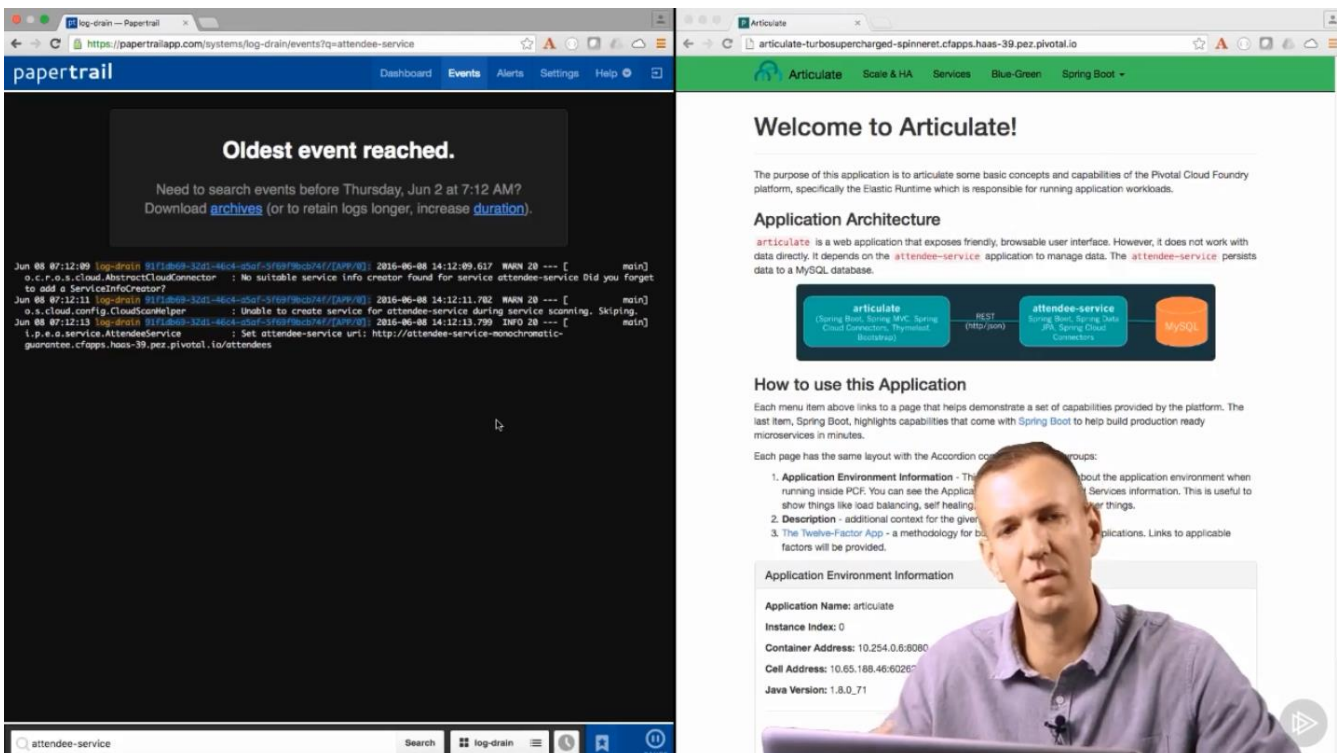
```
DROBERTS-MBPRO:articulate droberts$ cf restart articulate
Stopping app articulate in org dave / space dev as droberts@pivotal.io...
OK

Starting app articulate in org dave / space dev as droberts@pivotal.io...

0 of 1 instances running, 1 starting
```

The logs are now flowing into PaperTrail in near real-time.



And we can do specific searches on our logs. We can also use **Splunk** instead of PaperTrail.

Log Drain

Recap

