SpringOne Platform

Cloud Native for Spring Boot Devs

Thomas Gamble - The Home Depot

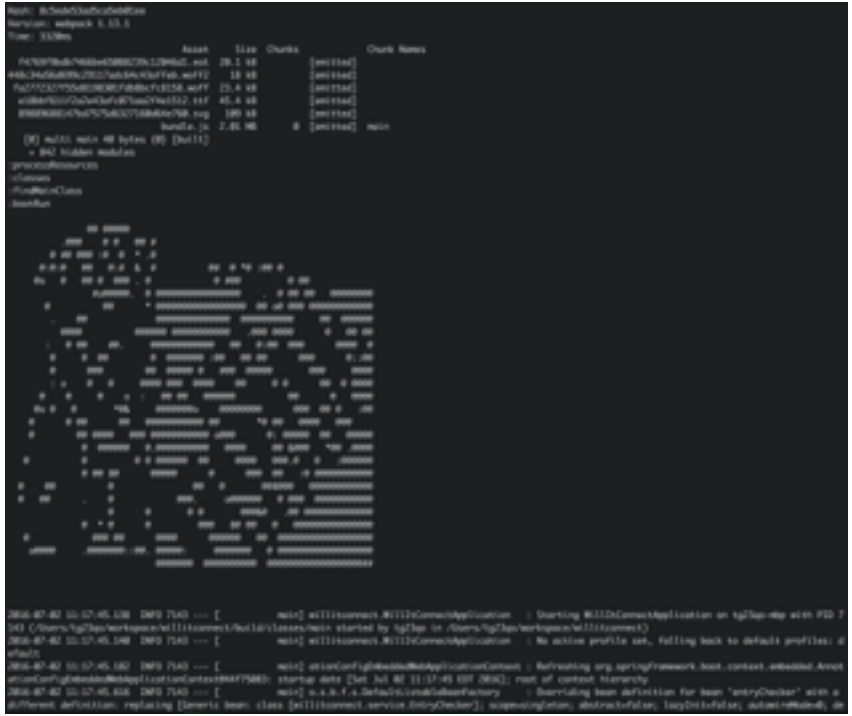
@gamtho

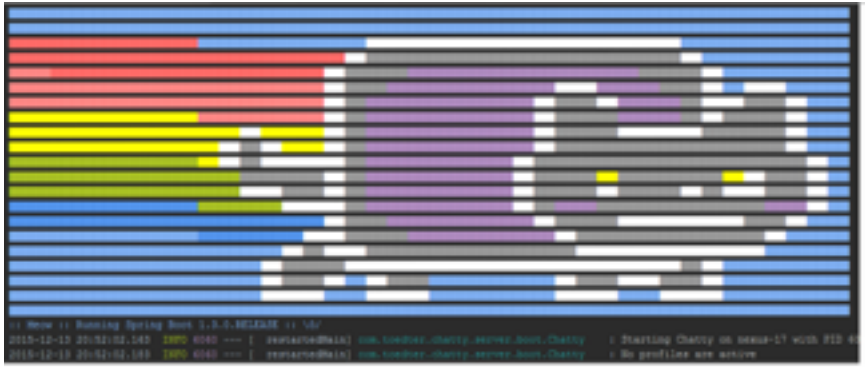
Pivotal

Factor 0 - Custom Banners

- Two easy options
  - Make/find your own ascii image
  - https://github.com/joshlong/bootiful-banners
  - ./src/main/resources/banner.txt
- Or with Spring boot 1.4
  - ./src/main/resources/banner.jpg

Gratuitous Banner Examples





Codebase: Using git doesn't mean you're doing it right

xkcd.com/1296/

xkcd.com/1597/

Factor 1 - Codebase

- One codebase, one application
  - tracked in revision control
  - one-to-one correlation between codebase and app
  - factor shared code into a library and use as a dependency

Actuator fun

```
    dependencies {
        classpath("org.springframework.boot:spring-boot-gradle-plu
        classpath("com.moowork.gradle:gradle-node-plugin:0.13")
        classpath("com.homedepot.gitprops:GitPropsPlugin:1.0.8")
    }

}

apply plugin: 'java'
apply plugin: 'eclipse'
apply plugin: 'idea'
apply plugin: 'spring-boot'
apply plugin: 'com.moowork.node'
apply plugin: 'com.homedepot.gitprops'
```

GET http://localhost:8080/info

Authorization Headers Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (4) Tests

Pretty Raw Preview JSON


```
1  {
2    "git": {
3      "commit": {
4        "message": "dependency bump",
5        "time": "Mon Jul 18 08:55:56 EDT 2016",
6        "id": "629bf29fd83344756a3fad8cb475973f9a6f6517",
7        "user": "gambtho"
8      },
9      "branch": "sb-1.4"
10   }
11 }
```


one of many other options - https://github.com/n0mer/gradle-git-properties

http://i.memecaptain.com/gend_images/suHPcQ.jpg

Factor 2 - Dependencies

- Make no assumptions about what will be provided
  - Explicitly declare
  - Bring what you need

- Relationship between app and runtime should be abstracted and isolated.
  - Embed server/container in the release artifact
  - Combined with release artifact by the platform (buildpack)

Explicit Version?

1. willitconnect: vim m

resources: vim applic... ⌘1 | willitconnect: vim ma... ⌘2

manifest.yml

```yaml
---
applications:
- name: willitconnect
  memory: 1G
  path: build/libs/willitconnect-1.0.4.jar
  buildpack: https://github.com/cloudfoundry/java-buildpack#v3.7.1
```

```
Downloading build artifacts cache...
Downloaded build artifacts cache (44.7M)
Staging...
-----> Java Buildpack Version: v3.7.1 | https://github.com/cloudfoundry/java-buildpack#8821e85
-----> Downloading Open Jdk JRE 1.8.0_91-unlimited-crypto from https://java-buildpack.cloudfoundry.org/openjdk/trusty/x86_64/openjdk-1.8.0_
-unlimited-crypto.tar.gz (1.4s)
       Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.1s)
-----> Downloading Open JDK Like Memory Calculator 2.0.2_RELEASE from https://java-buildpack.cloudfoundry.org/memory-calculator/trusty/x86_
/memory-calculator-2.0.2_RELEASE.tar.gz (0.0s)
       Memory Settings: -Xmx768M -XX:MaxMetaspaceSize=104857K -Xss1M -Xms768M -XX:MetaspaceSize=104857K
-----> Downloading Spring Auto Reconfiguration 1.10.0_RELEASE from https://java-buildpack.cloudfoundry.org/auto-reconfiguration/auto-recon
uration-1.10.0_RELEASE.jar (0.0s)
Exit status 0
Staging complete
Uploading droplet, build artifacts cache...
Uploading build artifacts cache...
Uploading droplet...
Uploaded build artifacts cache (90.1M)
```

Or No?

- Just supplied the jar, did not supply an execution environment
- The platform produces a release -- combines the JAR file with the current config

```
isitup: vim manifest.y...  ⌘1    ~/workspace/willi...  ⌘2
manifest.yml
---
applications:
- name: IsItUp
  memory: 512M
  instances: 1
  buildpack: https://github.com/cloudfoundry/java-buildpack.git
  path: ./build/libs/isitup-0.0.1.jar
```

```
Downloading build artifacts cache...
Downloaded build artifacts cache (45.4M)
-----> Java Buildpack Version: 7a37ff3 | https://github.com/cloudfoundry/java-buildpack.git#7a37ff3
-----> Downloading Open Jdk JRE 1.8.0_91-unlimited-crypto from https://java-buildpack.cloudfoundry.org/openjdk/trusty/x86_64/openjdk-1.8.0_91
-unlimited-crypto.tar.gz (3.0s)
       Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.2s)
-----> Downloading Open JDK Like Memory Calculator 2.0.2_RELEASE from https://java-buildpack.cloudfoundry.org/memory-calculator/trusty/x86_64
/memory-calculator-2.0.2_RELEASE.tar.gz (0.0s)
       Memory Settings: -Xms317161K -XX:MetaspaceSize=64M -Xss228K -Xmx317161K -XX:MaxMetaspaceSize=64M
-----> Downloading Spring Auto Reconfiguration 1.10.0_RELEASE from https://java-buildpack.cloudfoundry.org/auto-reconfiguration/auto-reconfig
uration-1.10.0_RELEASE.jar (0.1s)
Exit status 0
Staging complete
Uploading droplet, build artifacts cache...
Uploading build artifacts cache...
Uploading droplet...
```

Factor 3 - Configuration

- Config and Credentials should be externalized from Code
  - Declared
  - Inserted at runtime
- Don’t store your credentials in your code or your repo
- Store configuration in environment variables or services (cups), easy to change between deploys
- Independently managed for each deploy

Super Complex Examples - Manage Environment and Dependencies

```yaml
##spring:
##  profiles: default
security:
  oauth2:
    client:
      clientId: ${clientId}
      clientSecret: ${clientSecret}
      accessTokenUri: https://github.com/login/oauth/access_token
      userAuthorizationUri: https://github.com/login/oauth/authorize
      clientAuthenticationScheme: form
    resource:
      userInfoUri: https://api.github.com/user

---
spring:
  profiles: cloud
spring.jpa.database-platform: org.hibernate.dialect.PostgreSQLDialect
spring.jpa.generate-ddl: true
```

16 lines (13 sloc) | 314 Bytes

```yaml
---
recipient: nobody@example.com

spring:
  mail:
    host: localhost
    port: 25

---
spring:
  profiles: cloud
  mail:
    host: ${vcap.services.smtp-service.credentials.hostname}
    username: ${vcap.services.smtp-service.credentials.username}
    password: ${vcap.services.smtp-service.credentials.password}
```

Or Config Server

```java
@Component
public class TaxClient {

    private String url;
    private String key;
    private RestTemplate restTemplate;

    @Autowired
    public TaxClient(@Value("${tax.token}") String key, @Value("${ta
        this.url = url;
        this.key = key;
        this.restTemplate = restTemplate;
    }
```

68 lines (61 sloc) | 1.54 KB

```yaml
tax:
  url: http://taxservice-qa.
  token:
```

Branch: master usom-tax / src / main / resources / bootstrap.yml

jcc8813 add version to /info endpoint

1 contributor

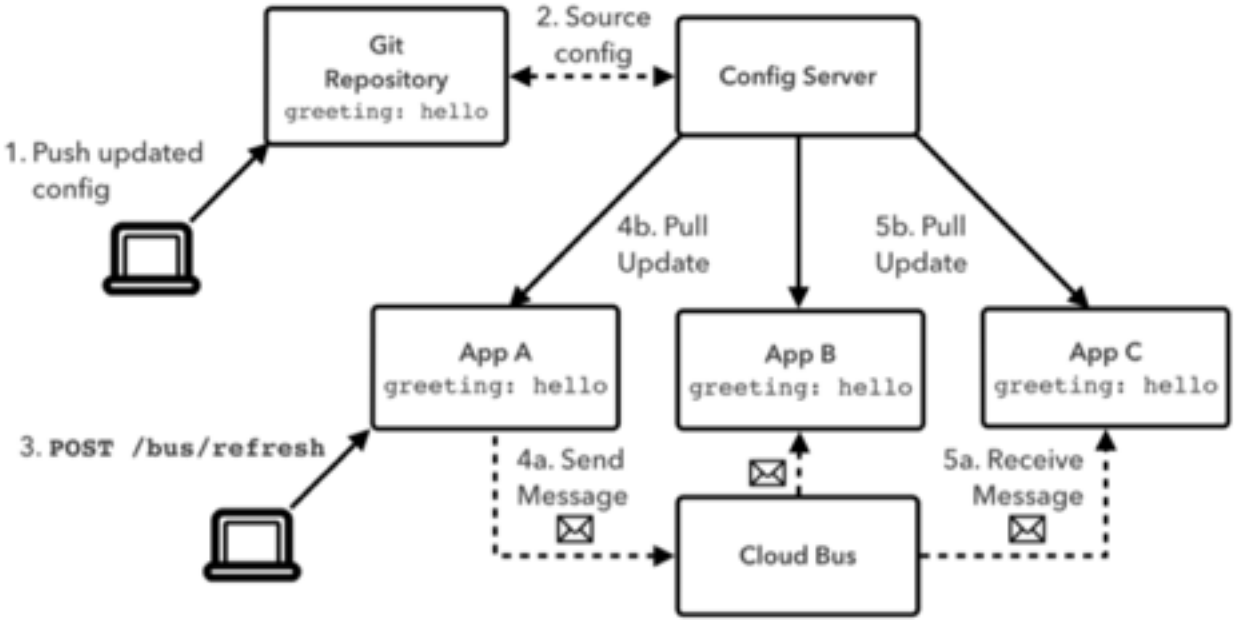
19 lines (17 sloc) | 331 Bytes

```yaml
spring:
  application:
    name: usom-tax
  cloud:
    config:
      uri: https://usom

info:
  build:
    name: usom-tax
    version: ${project.name}-${project.version}

---
spring:
  profiles: prod
  cloud:
    config:
      enabled: true
      uri:
```

Config Server with Cloud Bus



Random Gradle Tip

- When using spring profiles, it's helpful to quickly switch between them. One option

```
+ bootRun {
+     String activeProfile =  System.properties['spring.profiles.active']
+     systemProperty "spring.profiles.active", activeProfile
+ }
```

- ./gradlew -Dspring.profiles.active=cloud bootRun

Factor 4: Backing Services

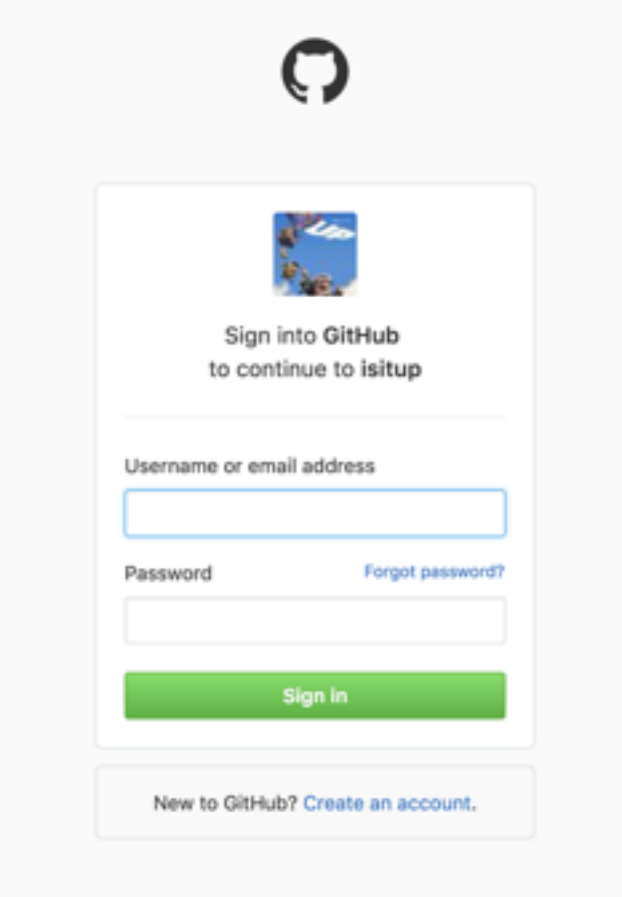
- Any service your application relies on for its functionality
  - data stores, messaging, caches
- Declare a need for a backing service, but let the runtime bind it
  - Should be possible to attach and detach without requiring an application deploy
  - Code should make no distinction between local and 3rd party services.

Auth as a backing service

```
dependencies {
        compile('org.springframework.boot:spring-boot-starter-a
        compile('org.springframework.boot:spring-boot-starter-d
        compile('org.springframework.boot:spring-boot-starter-d
    compile('org.springframework.security.oauth:spring-security
        compile("com.h2database:h2")
        compile('postgresql:postgresql:9.1-901-1.jdbc4')
    testCompile('org.springframework.boot:spring-boot-starter-t
```

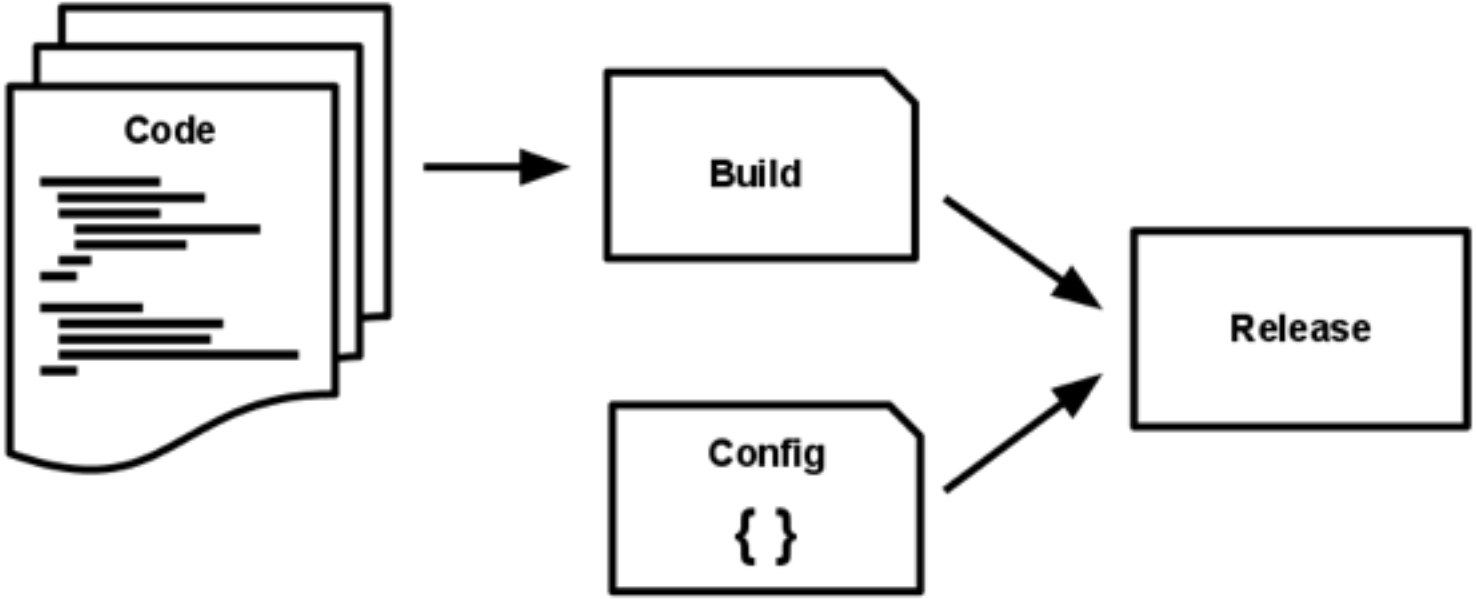
```
@SpringBootApplication
@EnableOAuth2Sso
@Controller
public class IsItUp extends WebSecurityConfigurerAdapter {

    @Bean
    public AuthoritiesExtractor authoritiesExtractor(OAuth
        return map -> {
            String url = (String) map.get("organizations_u
            @SuppressWarnings("unchecked")
            List<Map<String, Object>> orgs = template.getF
```



Factor 5 - Build Release Run

- build (initiated by dev), release (platform/tool), run (platform)
- One build -- many deploys.

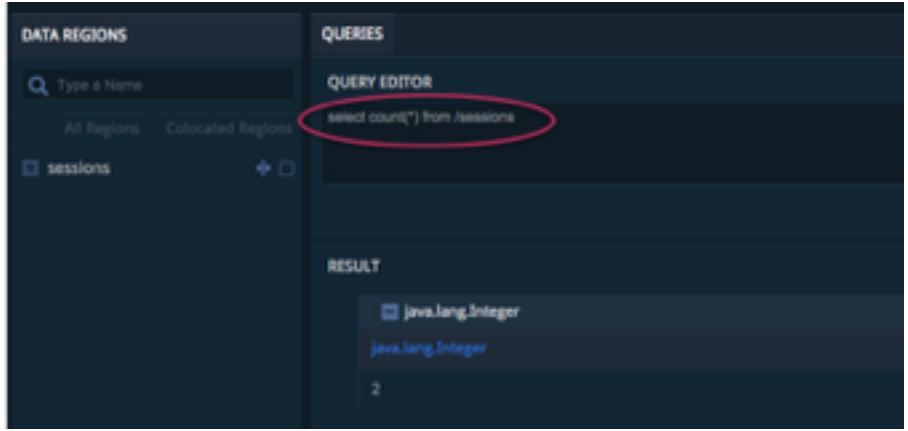


Factor 6 - Process Management

- One application, one process
  - Stateless
  - Share nothing

- Any data needed should be stored in a stateful backing service
  - session caching
  - blob storage
  - database

Session State Caching - Gemfire




```
dependencies {
    compile("org.springframework.boot:spring-boot-starter-data-gemfire")
```

Factor 7 - Port Binding

- The port is provided by the env and fed to the app via the start command
- Export services via port binding
- Applications are self contained — not injected into an external app server
- Allows applications to act as backing service for other applications
- Many ways to configure this
  - `docker run -p HOST:CONTAINER`
  - `var port = process.env.PORT || 3000;`
  - `buildpack magic / server.port=${port:8080}`
  - `http.ListenAndServe(":"+os.Getenv("PORT"), nil)`

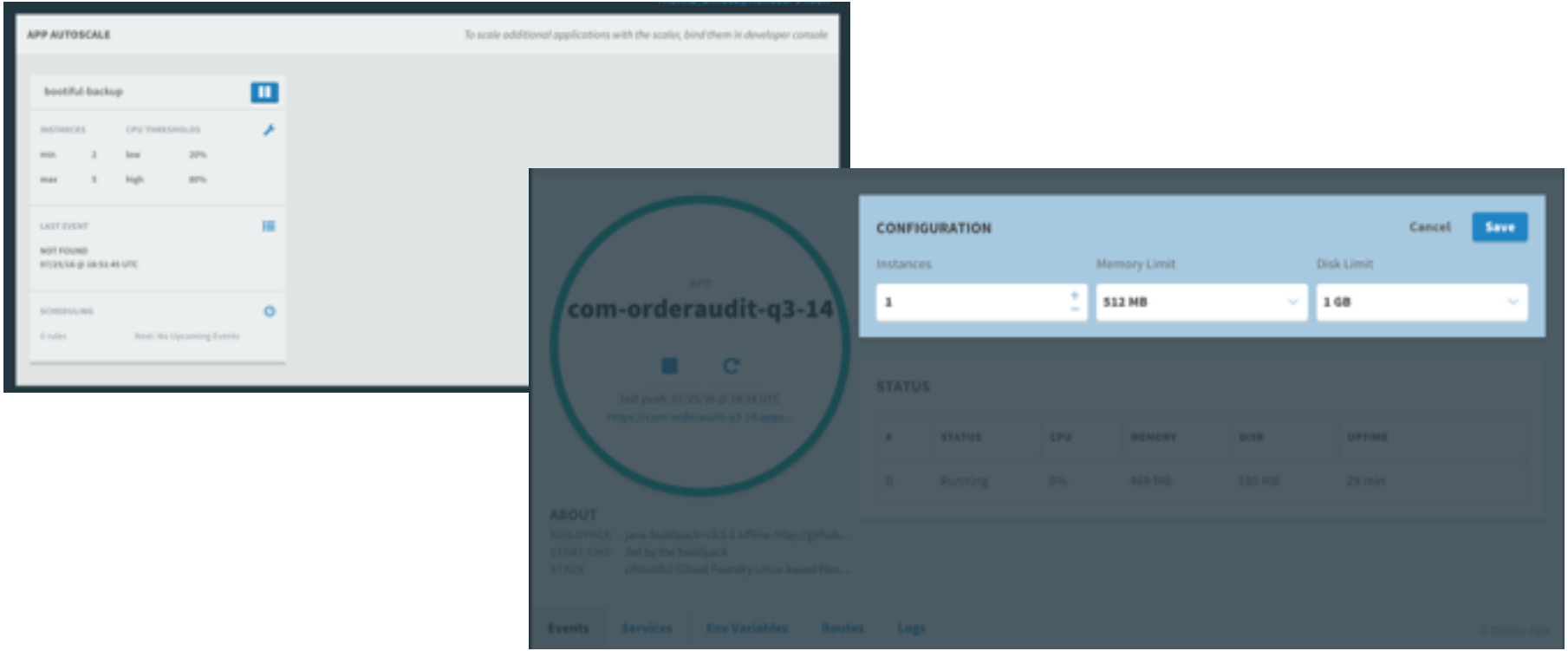
Factor 8 - Concurrency

- Scale out via the process model
  - multiple processes with distributed load
  - horizontal scale

Factor 9 - Disposability

- Running deployments should never be patched, but always disposed of.
- Minimize startup time
  - Backing services instead of in-memory cache
  - Scale quickly and easily
- Shutdown gracefully
  - Stop accepting new work, and let existing work finish
  - Or — push tasks to a queue
  - Release any locks or other resources

Scaling



Other scaling options

https://github.com/cloudfoundry-samples/cf-autoscaler - Matt Stine

Factor 10 - Dev/Prod Parity

- Minimize gap between dev and prod
- Declare what you need because you fail fast
  - Cups > env variables

Very long setup process

1. Download the latest version of PCF Dev CLI plugin from the Pivotal Network.

2. Unzip the downloaded zip file:

```
$ unzip pcfdev-VERSION-osx.zip
```

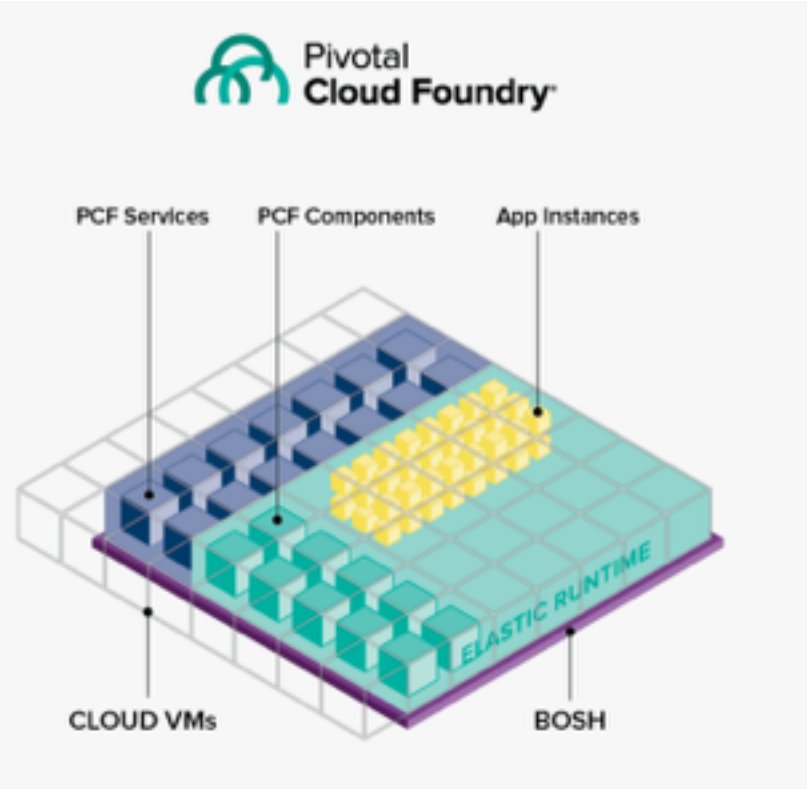
3. Install the PCF Dev plugin:

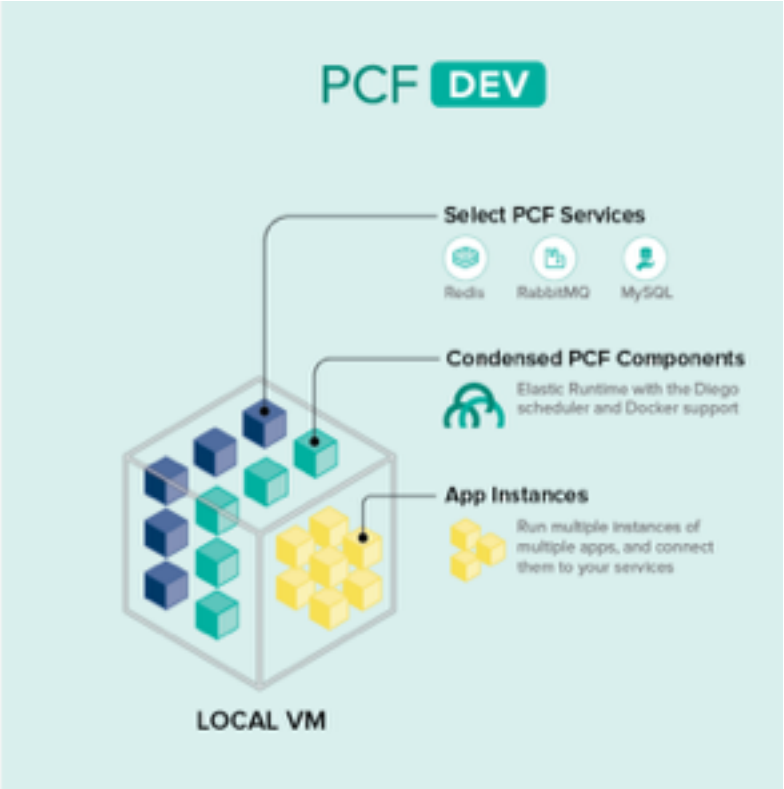
```
$ ./pcfdev-VERSION-osx
```

4. Start PCF Dev:

```
$ cf dev start
```

pretty picture - cf dev start





Factor 11 - Logging

- logs go to stdout and are a stream rather than a file
  - flow continuously as long as the app is running
- Stream should be captured by the execution environment, routed/stored/indexed as needed

Using stdout, doesn't mean be lazy

```
public void log(Enum<?> key, Object value)
{
        if(logBuilder!=null)
        {
                logBuilder.put(key, value);
        }
}

protected final LogBuilder put(String key, Object value)
{
    String valueStr = (value == null) ? "null" : value.to

    try {
        this.getMap().put(key, valueStr);
    }catch(Exception e)
    {
```

```
-]
@host: ap
@timestamp: 2016-07-21T17:42:39.564Z
attributes: { [+]
}
body: { [-]
  amount_to_auth: 146.96
  available_amount_for_auth: 10000.00
  com_event_type: Authorize
  context: COMAuthorization
  customer_order_number: W3702211119
  http_response: {'Orders':{'Order':{'DocumentType':'
d8ee864e362'}},'ExtnHostOrderReference':'W3702211119','
TranType':'AUTHORIZATION','Authorizat
T17:42:39','AllocatedAutorization':146.96,'PaymentStat
  lcp: q3
```

So that…

New Search

Save

```
index=qa_app_logs "com_event_type":"Bill" AND "failure_reason" AND NOT(ORDER_RECALL_FAILED)
earliest=-1d | dedup id | rename "body.customer_order_number" as "ORDER_ID", "body.bill_how" as
"BILL_HOW","body.retryable" as "RETRY_ABLE", "body.failure_reason" AND "FAILURE_REASON" | table _time,
ORDER_ID , BILL_HOW, RETRY_ABLE, FAILURE_REASON
```

Last 15 minu

✓ 3 events (7/21/16 1:59:40.000 PM to 7/22/16 1:59:41.154 PM)

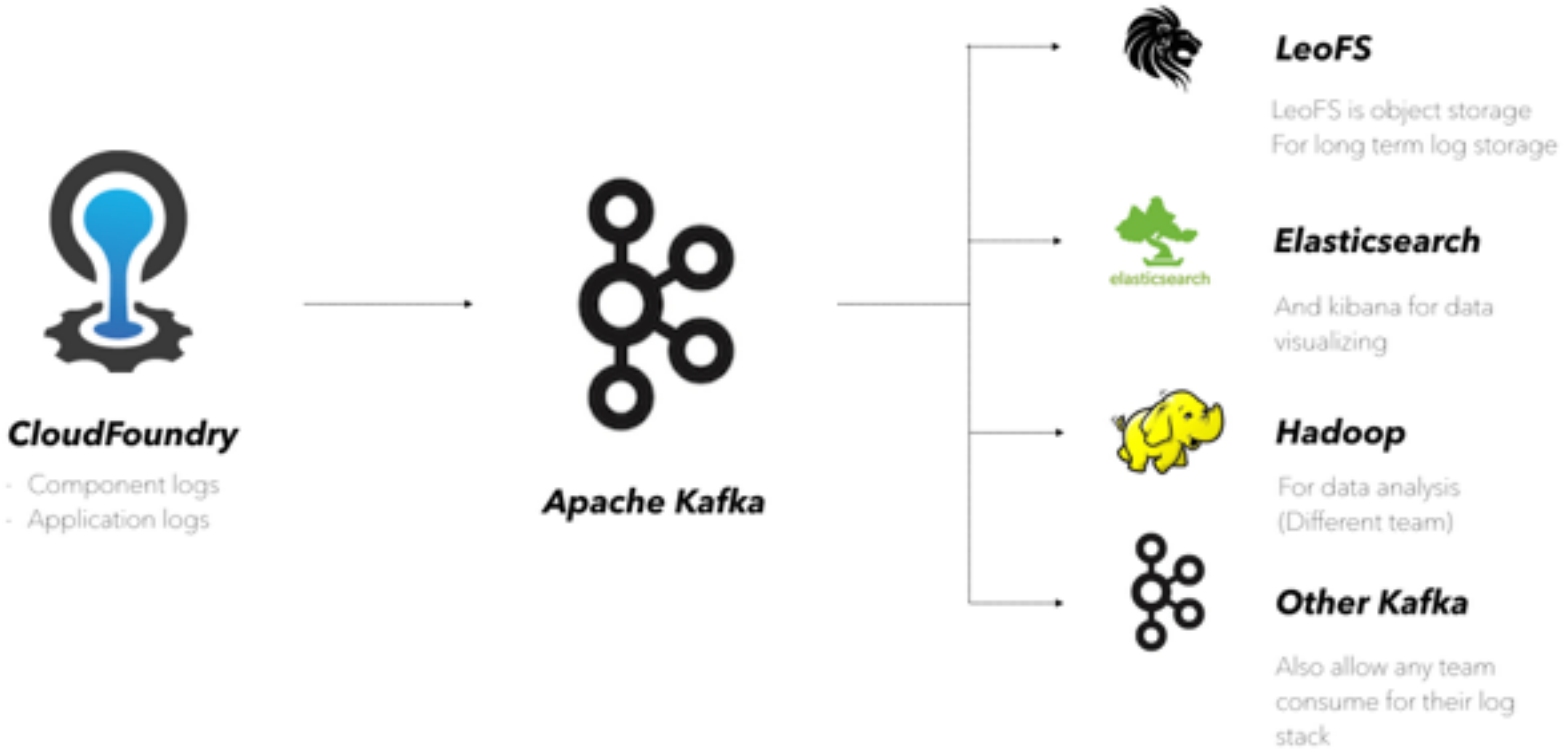
Job

Events | Patterns | Statistics (3) | Visualization

20 Per Page | Format | Preview

_time	ORDER_ID	BILL_HOW	RETRY_ABLE	FAILURE_REASON
2016-07-22 11:52:00.610	W88634688804		false	APPLICATION_ERROR: [Unable to build ModelPOSlog for order: W88634 Exception: No transactional quantity available to be processed on line - 0.00]
2016-07-22 08:01:53.589	W410540089	ForceBill	true	CREDIT_SERVICES_FAILURE: [Gift card processing - Network offline]
2016-07-22 07:51:52.603	W410540089	ForceBill	true	CREDIT_SERVICES_FAILURE: [Gift card processing - Network offline]

Log Stream Options



http://techblog.rakuten.co.jp/2016/01/28/rakuten-paas-kafka/

Factor 12 - Operational Tasks

- One-off or admin process
  - should be run in an identical environment as the app
  - should run against a release
  - stored in the same codebase and use the same config as any other process in the release
  - stored and ships with the application

Automate all the things....and Fail Fast

- compile "org.flywaydb:flyway-core:4.0.3"

application.yml /Users/tg23qo/Desktop

```
# FLYWAY (FlywayProperties)
flyway.baseline-description= #
flyway.baseline-version=1 # version to start migration
flyway.baseline-on-migrate= #
flyway.check-location=false # Check that migration scripts location exists.
flyway.clean-on-validation-error= #
flyway.enabled=true # Enable flyway.
```

References

- http://cloudfactor.io/factor/twelve/ - Josh McKenty
- http://pivotal.io/beyond-the-twelve-factor-app - Kevin Hoffman
- http://12factor.net/ - Adam Wiggins
- https://github.com/joshlong/bootiful-banners - Josh Long
- http://www.slideshare.net/SpringCentral/12-factor-cloud-native-apps-for-spring-developers - Cornelia Davis and Josh Kruck
- http://www.slideshare.net/mariofusco/comparing-different-concurrency-models-on-the-jvm - Mario Fusco
- http://toedter.com/2015/12/13/creating-colorful-banners-for-spring-boot-applications/ - Kai Todter
- http://www.infoworld.com/article/2925047/application-development/build-self-healing-distributed-systems-with-spring-cloud.html - Matt Stine
- https://spring.io/blog/2015/01/12/the-login-page-angular-js-and-spring-security-part-ii - Josh Long

SpringOne Platform

Learn More. Stay Connected.

@gambtho on twitter and github

@springcentral
spring.io/blog

@pivotal
pivotal.io/blog

@pivotalcf
http://engineering.pivotal.io