Pivotal CF

Building Cloud ready apps

What Platform Operators should know

Pivotal

The limitations of traditional apps

Scalability

Inter-Dependency

Platform Specificity

Location Specificity

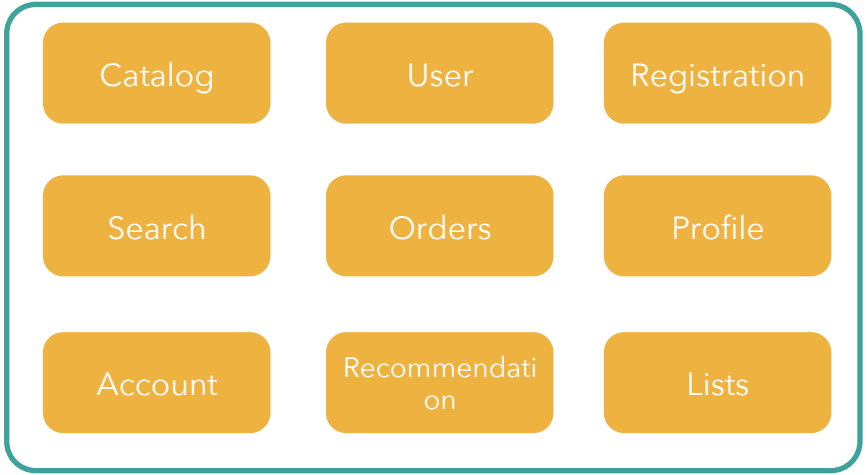
Resiliency

Traceability / Logging

Pivotal

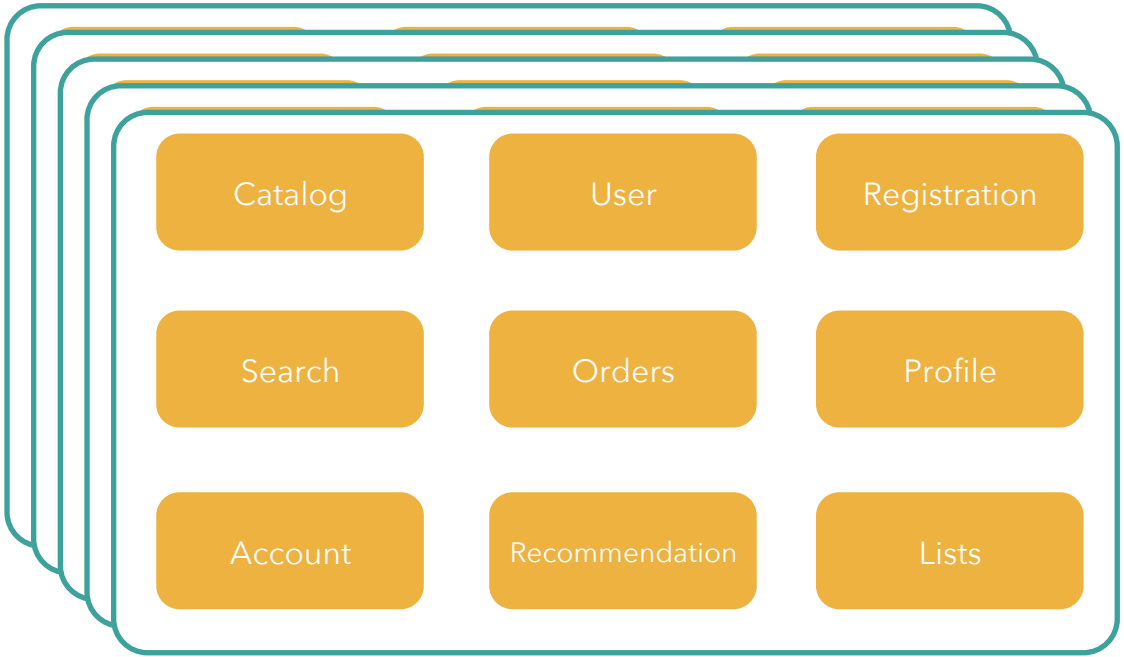
Scalability

- Traditional applications have been modeled as monolithic due an easy deployment model
- Several services are combined into one massive single application
- This model leads to a poor use of resources when it comes to scaling out your application
- Greedy components steal resources from others that are collocated with them



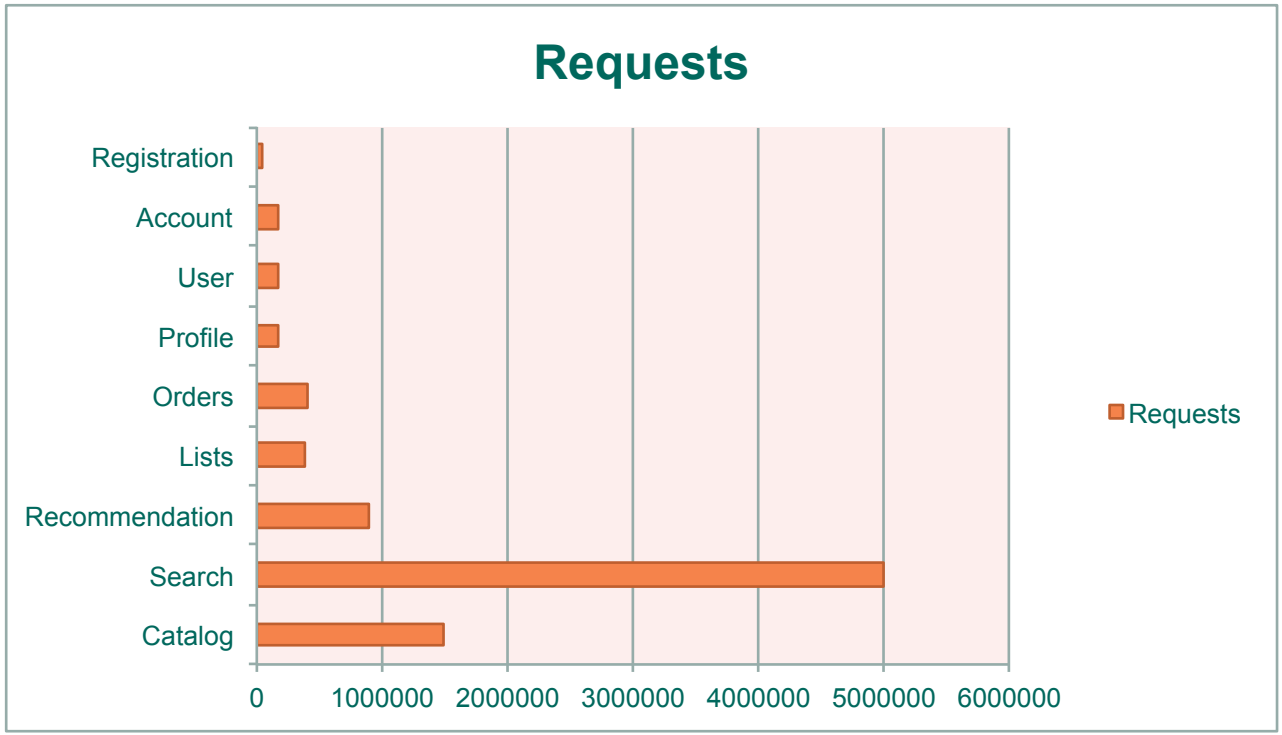
Pivotal

Scaling monolithic apps



Pivotal

How you should scale



Location Specificity : Writing to disk

- Applications often need to write to disk, this includes form uploads with binary data, or content in most CMS systems
- Containers are short lived and not guaranteed to be executed on the same hardware every time they are needed to restart. Depending on a local file system is a big lock dependency some applications impose on the runtime
- This is one of the first issues to question or address when starting a conversation around new applications or selecting candidates to execute in PCF
- Some CMS vendors support usage of a service such as S3 to be the persistent mechanism of choice

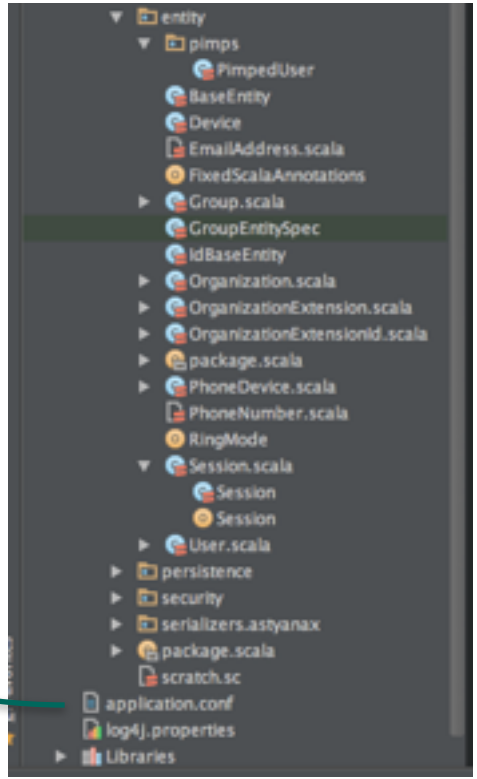
Pivotal

Location Specificity : service locations

```
Cache.hosts=10.68.27.41,10.68.27
.42

#naming does not help either
Cache.hosts=cacheserver1,
cacheserver2
```

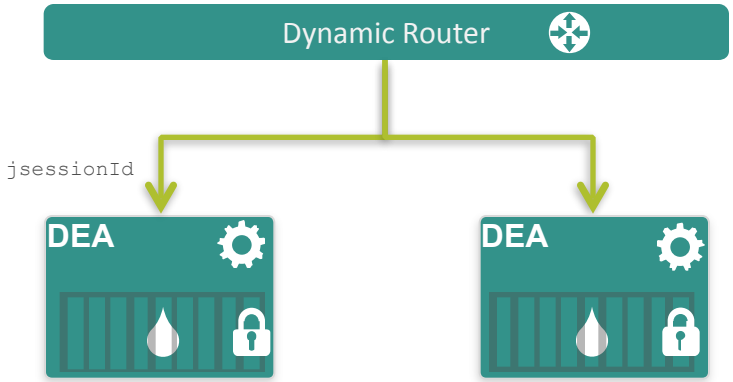
Properties file deployed
With application



Pivotal

Location Specificity : Sticky sessions

- Applications that rely on session stickiness are subject to lead to unexpected behavior in case of failure of the node that contains that data
- Most web applications that use sessions do not have a replication strategy in place
- Your application availability is subject to a single node (even if you LB the load over other nodes)
- Some app servers use P2P session replication which can be very costly on large deployments



Pivotal

The limitations of traditional apps

Scalability

Inter-Dependency

Platform Specificity

Location Specificity

Resiliency

Traceability / Logging

Pivotal

Resilience

- No graceful shutdown in case of catastrophic events
- No recovery when the process executing the application dies
- No horizontal and automatic scale under heavy load
- No fallback mechanisms when dependent services are broken

Pivotal

The limitations of traditional apps

Scalability

Inter-Dependency

Platform Specificity

Location Specificity

Resiliency

Traceability / Logging

Pivotal

Traceability / Logging

- Traditional apps usually write logs to files on disk
- When you have a cluster composed of several instances, it becomes really hard to trace log files spread across different locations

Pivotal

3rd Platform Strategies

Pivotal

Better 3rd Platform Strategies

Local Disk Storage

Embedded Services

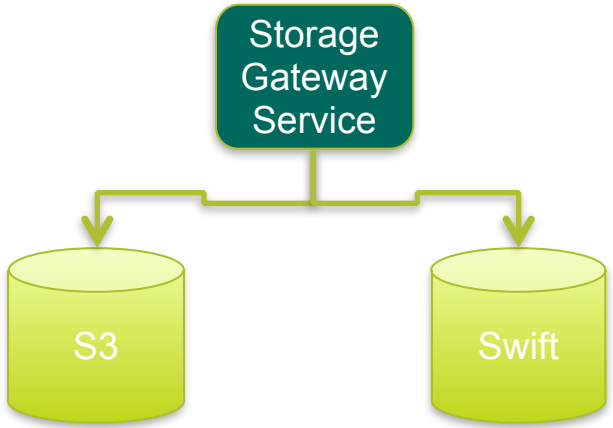
Session Replication

Logging

Pivotal

Local disk storage

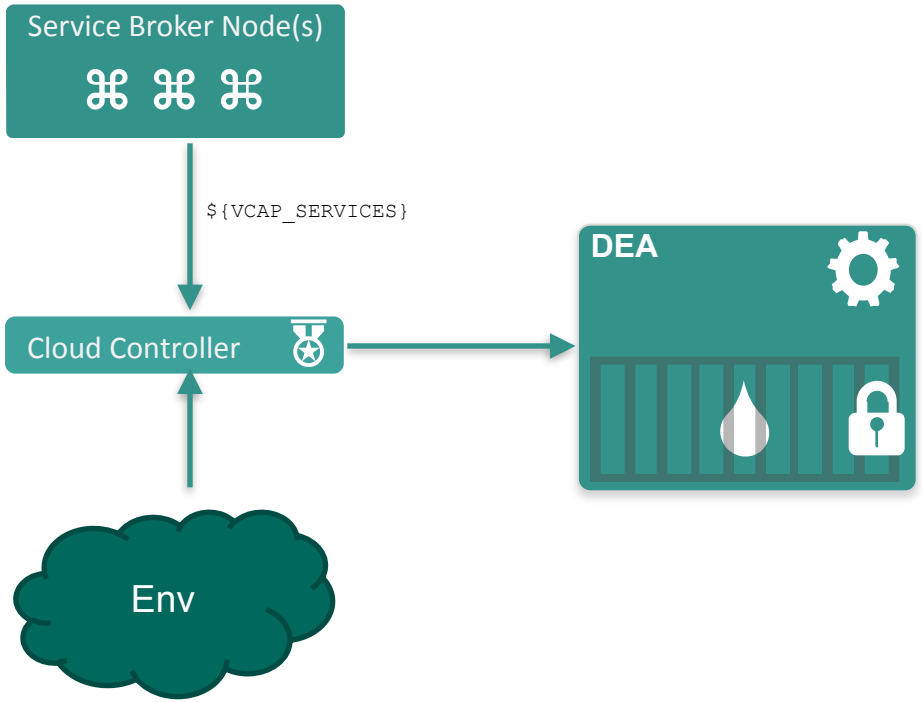
- Use an storage gateway service instead of relying on plain old file system access
- This abstracts the need of a local file system while giving your application a flexible mechanism to rely depending on it's runtime (local on your dev, S3 or swift on prod)



Pivotal

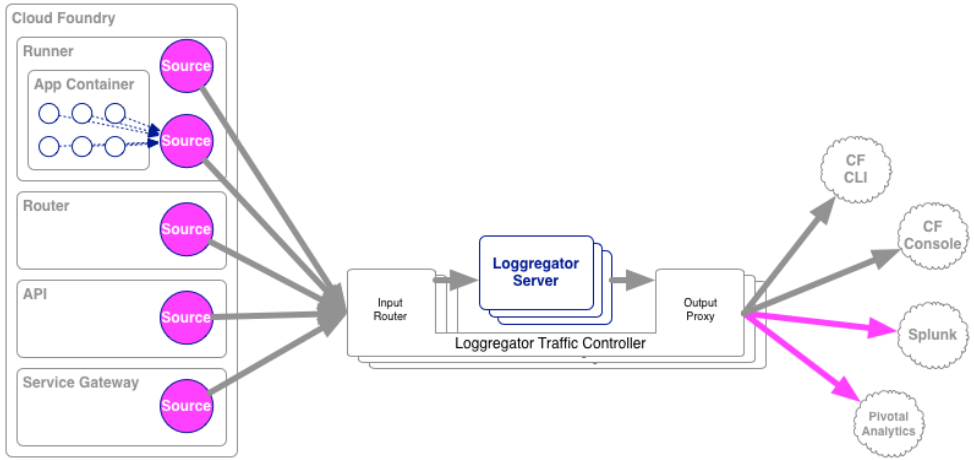
Embedded services

- Let the runtime inject any service reference through environment variables
- Reduces **location specificity** of the application
- Allows **disposability** of containers
- Runtime should **inject** any **variable** into the container



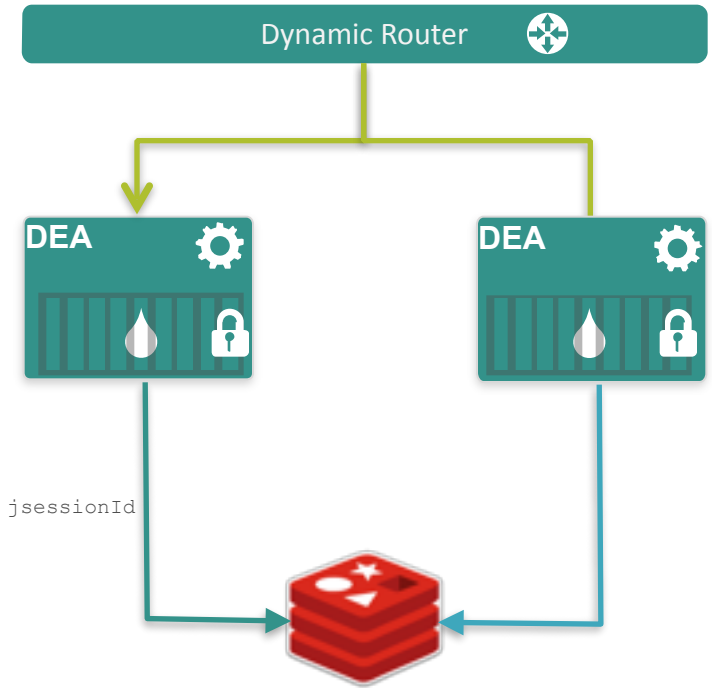
Logging

- Do not write to log files
- Output logs to console and use a syslog drain to collect all logs



Session replication

- The java buildpack supports redis session replication
- Sessions are now stored on redis, and in case of a node failure, the client state can still be found on a second node



Pivotal

3rd Platform Realities

Pivotal

Paying for your lunch…

- Significant Operations Overhead
- Substantial DevOps Skills Required
- Implicit Interfaces
- Duplication of Effort
- Distributed System Complexity
- Asynchronicity is Difficult!
- Testability Challenges

Pivotal

Paying for your lunch…

- **Significant Operations Overhead**
- **Substantial DevOps Skills Required**
- Implicit Interfaces
- Duplication of Effort
- **Distributed System Complexity**
- Asynchronicity is Difficult!
- **Testability Challenges**

Pivotal

Significant Operations Overhead

- Mitigate polyglot language/environment provisioning complexity via CF Buildpacks
- Mitigate failover and resilience concerns via CF Scale, CF Health Monitor, and future CF App AZ's (http://blog.gopivotal.com/cloud-foundry-pivotal/products/the-four-levels-of-ha-in-pivotal-cf)

Pivotal

Significant Operations Overhead

- Mitigate polyglot language/environment provisioning complexity via CF Buildpacks
- Mitigate failover and resilience concerns via CF Scale, CF Health Monitor, and future CF App AZ's (http://blog.gopivotal.com/cloud-foundry-pivotal/products/the-four-levels-of-ha-in-pivotal-cf)
- Mitigate routing/load balancing and plumbing concerns via CF Router and CF Services
- High quality monitoring = CF BP agent-based tooling, future CF metric streams
- High quality operations infrastructure = CF BOSH!

Pivotal

Significant Operations Overhead

- Mitigate polyglot language/environment provisioning complexity via CF Buildpacks
- Mitigate failover and resilience concerns via CF Scale, CF Health Monitor, and future CF App AZ's (http://blog.gopivotal.com/cloud-foundry-pivotal/products/the-four-levels-of-ha-in-pivotal-cf)
- Mitigate routing/load balancing and plumbing concerns via CF Router and CF Services
- High quality monitoring = CF BP agent-based tooling, future CF metric streams
- High quality operations infrastructure = CF BOSH!
- Robust release/deployment automation = CF API, scriptable CF CLI, Maven/Gradle Plugins, Strong Cloudbees/Jenkins partnerships

Pivotal

Substantial DevOps Skills Required

- This is a **Good Thing™** in any architecture!
- CF keeps your microservices up and available (and your monoliths too!)
- CF = development and production parity!
- Polyglot persistence without all the fuss: CF BOSH and Service Brokers

Pivotal

Distributed System Complexity

- Agreed: Microservices imply distributed systems.
- All of the CF platform features we've discussed help to mitigate these concerns:
  - latent/unreliable networks
  - fault tolerance
  - load variability

Pivotal

Testability Challenges

- With CF, it is ***NOT*** difficult to recreate environments in a consistent way for either manual or automated testing!
- Idiomatic Microservices involves placing less emphasis on testing and more on monitoring
  - Not sure where this idea comes from…
  - CF is an enabler of both!

Pivotal

A NEW PLATFORM FOR A NEW ERA