LLamasoft®

Supply Chain By Design

The Evolution of the DevOps Pipeline: To K8s and Beyond!

Josh Lynn – Senior Cloud Engineer - LLamasoft

Josh Lynn

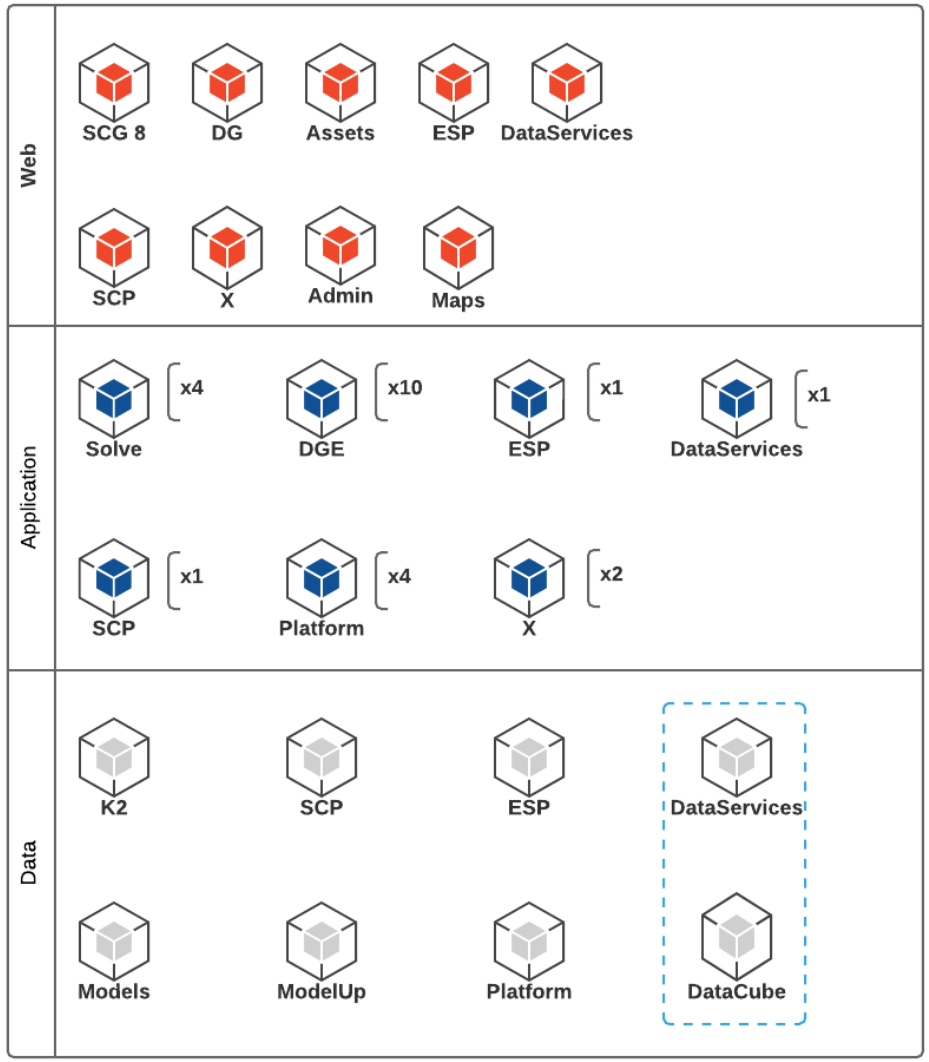
- 15 years in Industry
  - Thomson Reuters
  - Hooklogic (Acquired by Criteo)
  - Criteo
  - LLamasoft

Agenda

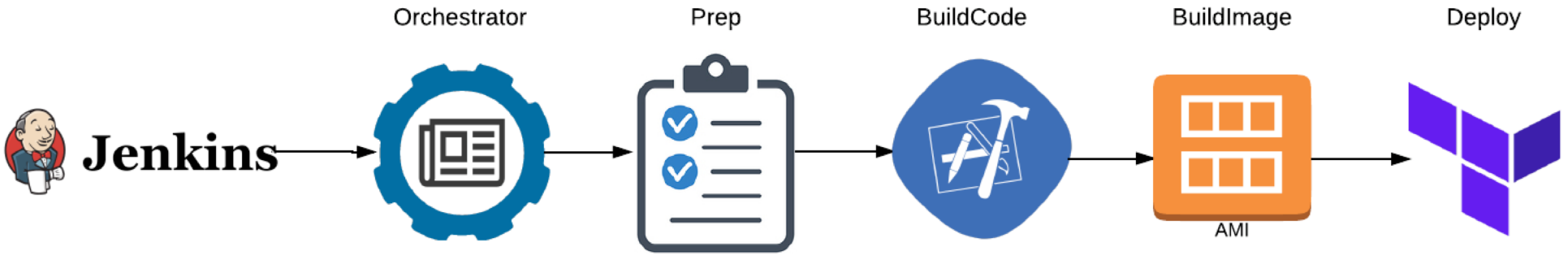
- Introduction
- By the Numbers
- Packer, Chef, Terraform, Sequencer, Consul
- Jenkins Pipelines
- Kops
- Kubernetes (K8s)
- K8s Lessons Learned
- What's Next?

By The Numbers

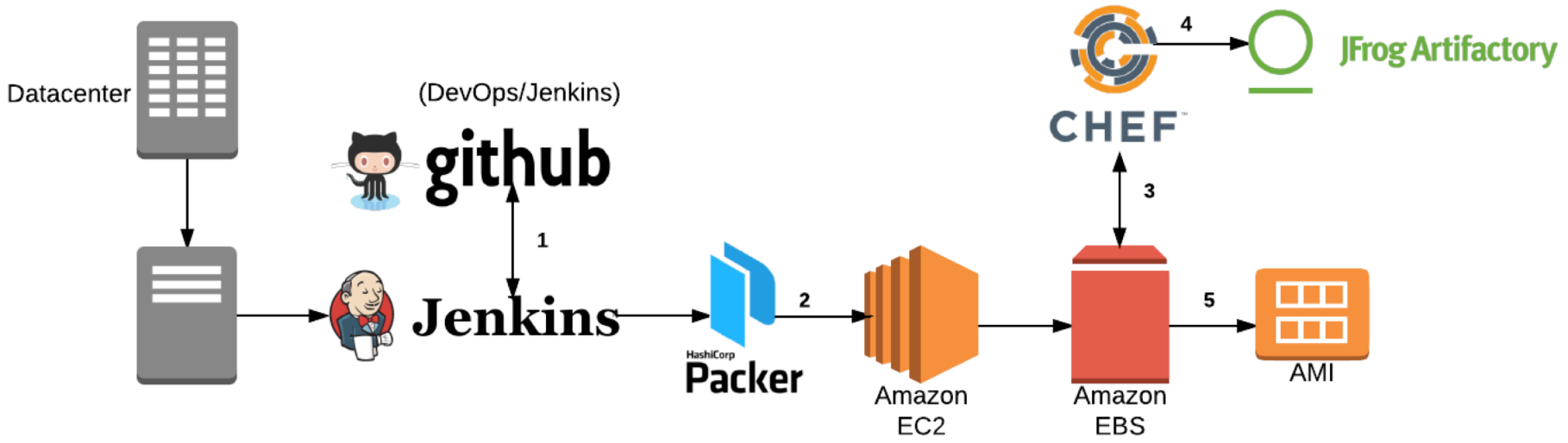
- 9 Websites
- 20+ Services
- 8 Databases
- 3 External Applications



The DevOps Pipeline



The DevOps Pipeline: BuildImage



The DevOps Pipeline: Packer

Packer is an open source tool for creating identical machine images for multiple platforms from a single configuration.

- Advantages
  - Lightweight, Highly performant
  - Runs on every major operating system
  - Creates images for multiple platforms in parallel (AWS, Vmware, Docker)

The DevOps Pipeline: Packer Tips & Tricks

I want to see more detail where the packer build is failing, ***what do I do?***

- Debug
  - If you want to step through the build process, use the –debug flag in your packer command:

```
packer build -debug app.json
```

- Logging
  - You can also add logging to have the packer output written to a specified location:
  - Note: A log will be generated by setting PACKER_LOG equal to anything but “”

```
export PACKER_LOG_PATH="/path/to/packer.log"
export PACKER_LOG=10
```


The DevOps Pipeline: Packer Tips & Tricks

DEMO

The DevOps Pipeline: Chef

Chef – An open-source tool used for configuration management.

- Advantages
  - Accelerating software delivery by allowing deployment of software within 1 hour of a commit.
  - Integrated testing framework (Test Kitchen)
  - Maintain a consistent configuration
    up the stack (Dev → QA → Staging → Prod)

The DevOps Pipeline: Chef

- Test Kitchen is a **testing** tool that can execute your CHEF recipes on one or more platforms in isolation. A driver plugin architecture is used which lets you run your recipes on various cloud providers and virtualization technologies.

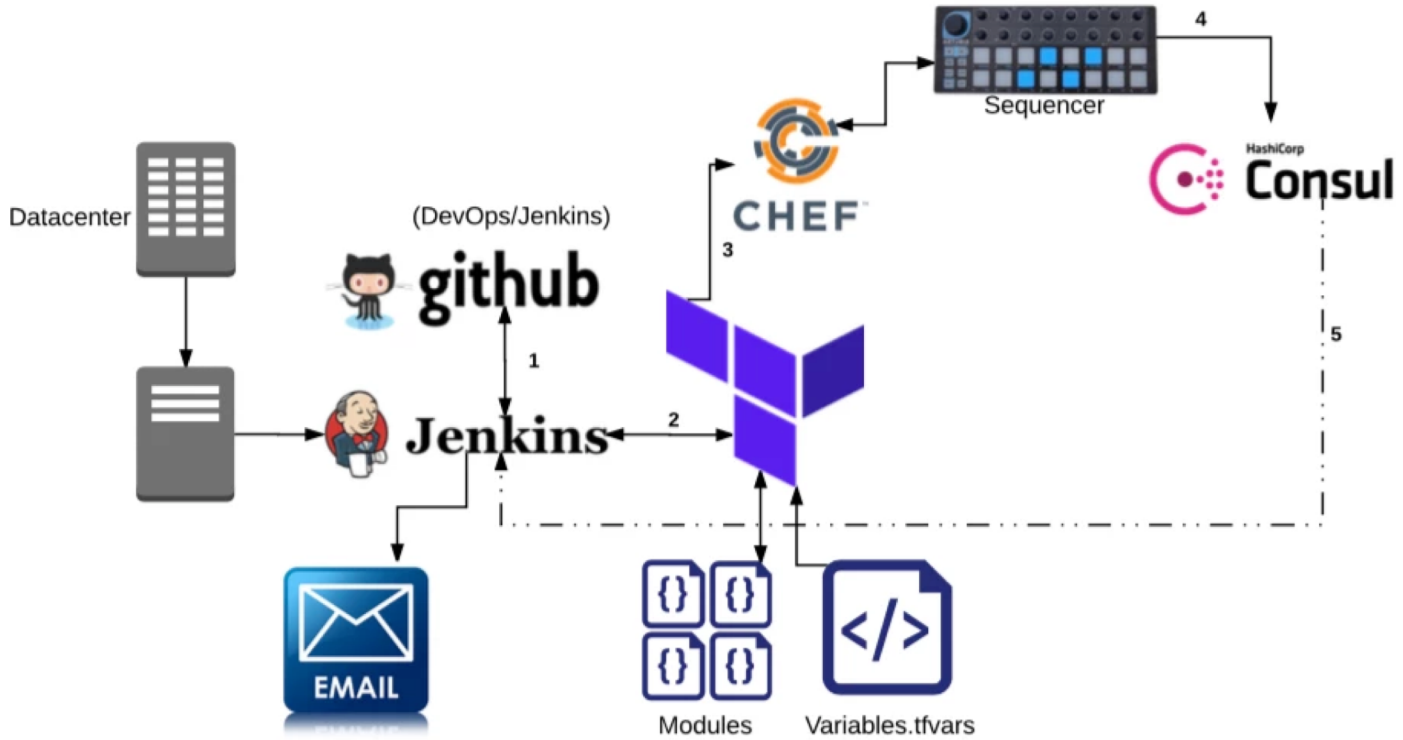
- Setup
  - Full Setup Tutorial

- Types
  - Integration (*ServerSpec* based on Rspec)
  - Unit *(ChefSpec* based on Rspec*)*
  - Most common is **Integration.**

The DevOps Pipeline: Chef

DEMO

The DevOps Pipeline: Deploy



The DevOps Pipeline: Terraform

Terraform enables you to safely and predictably create, change, and improve infrastructure.

- Advantages
  - Cloud platform agnostic (AWS, Azure, GCP)
    - *Well…sort of….*
  - Code Readability (Written in GO, mostly API calls)

The DevOps Pipeline: Sequencer

What is Sequencer?

- Sequencer is essentially a script written in Ruby that takes values in and makes GET requests to Consul to check status on various components being deployed by Terraform.
- Sequencer is used to provide visibility into the deployment of Terraform components during provisioning (deployment).

The DevOps Pipeline: Sequencer

```ruby
node['sequencer']['dependencies'].each do |dependency|
    # Updates value to let user know we're waiting
    http_request 'Set value to Waiting' do
        message                         "WAITING on #{dependency}"
        url                             "http://#{node['sequencer']['consul_address'
        action                          :put
    end

    ruby_block 'Check dependencies' do
        block do
            require 'net/http'
            require 'base64'
            require 'json'
            puts "#{dependency}"
            http = Net::HTTP.new('consul.management.<redacted>.com', 80)
            response = http.send_request('GET', "/v1/kv/sequencer/#{node['sequer
            puts response.body
            puts response.code
            json_response = JSON.parse(response.body)
            # You'll notice the 2nd operator for the comparison (base64 encoded
            # Beware of this gotcha if you choose to refactor this
            while json_response[0]["Value"] != Base64.encode64('FINISHED').strip
                http = Net::HTTP.new('consul.management.<redacted>.com', 80)
                response = http.send_request('GET', "/v1/kv/sequencer/#{node['se
                json_response = JSON.parse(response.body)
                sleep 5
            end
        end
    end
end
```


The DevOps Pipeline: Consul

Consul has multiple components, but as a whole, it is a tool for discovering and configuring services in your infrastructure. It provides several key features: Service Discovery: Clients of Consul can provide a service, such as api or mysql , and other clients can use Consul to discover providers of a given service.

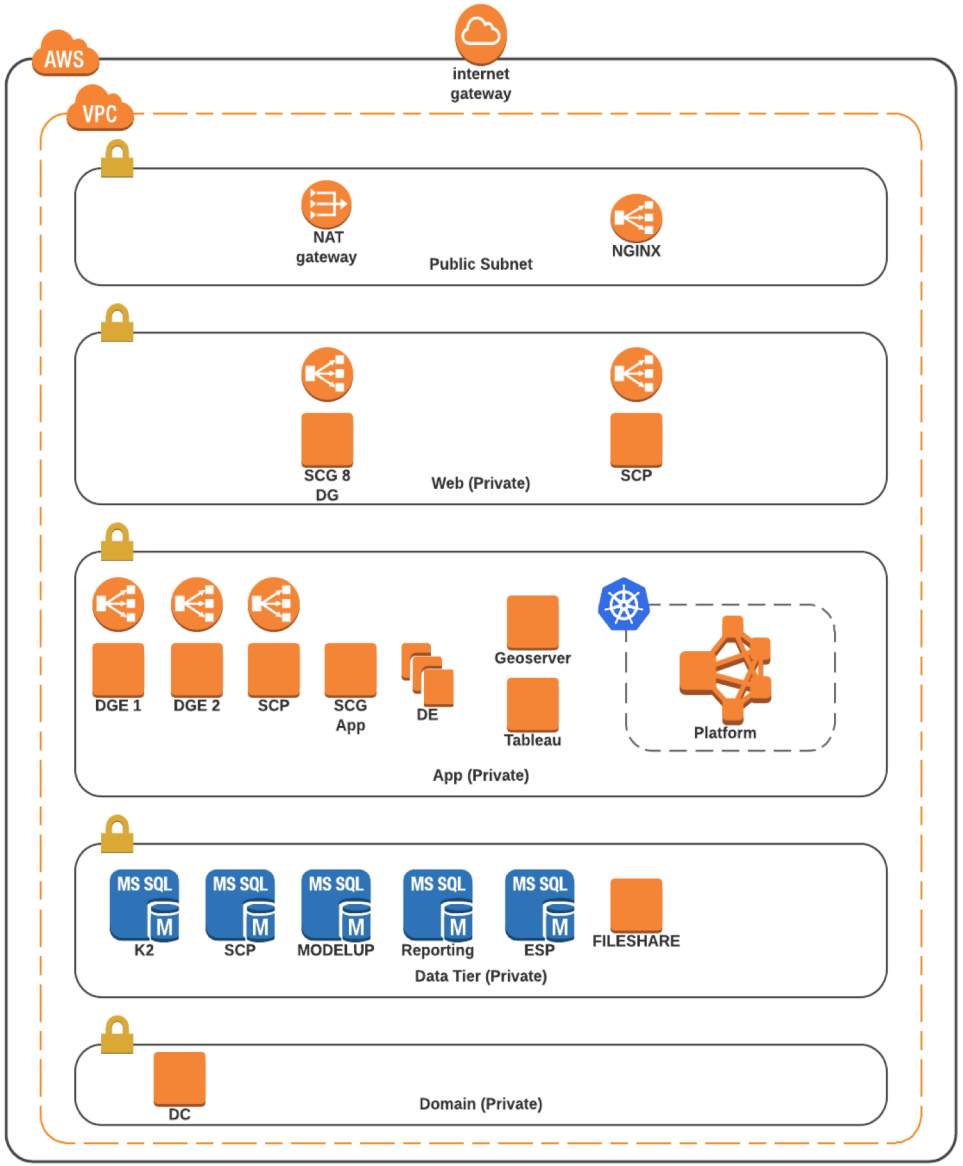
How do we use it?

- Consul is simply a key/value store for us to store terraform state and be able to query whether a component deploy is running, waiting or deployed.

The DevOps Pipeline: Consul


```
["convergence-terraform","convergence-terraform-env:CIDEV","convergence-terraform-env:FUUUUU","convergence-terraform-env:Josh
env:JoshSQLTest2","convergence-terraform-env:JoshTest","convergence-terraform-env:JoshTestProxy","convergence-terraform-env:K
env:MoarProxy","convergence-terraform-env:ProxyTest","convergence-terraform-env:ProxyTest2","convergence-terraform-env:ProxyT
terraform-env:QA","convergence-terraform-env:SQLTest","convergence-terraform-env:TownHall","convergence-terraform-env:ad_test
env:adtest2","convergence-terraform-env:adtest3","convergence-terraform-env:amdemts","convergence-terraform-env:artenv3","con
env:best-dev","convergence-terraform-env:bigfatfailure","convergence-terraform-env:build012","convergence-terraform-env:build
terraform-env:ci-dev","convergence-terraform-env:ci_dev","convergence-terraform-env:cidev","convergence-terraform-env:cidev80
terraform-env:da204","convergence-terraform-env:dabigtest","convergence-terraform-env:datest127","convergence-terraform-env:da
terraform-env:datesting","convergence-terraform-env:dev","convergence-terraform-env:dev006","convergence-terraform-env:dev3",
env:djw312","convergence-terraform-env:djw313","convergence-terraform-env:djw314","convergence-terraform-env:djw315","converge
env:dreamofci","convergence-terraform-env:dropthebass","convergence-terraform-env:dwq123","convergence-terraform-env:env001",
env:env003","convergence-terraform-env:env004","convergence-terraform-env:env005","convergence-terraform-env:env007","converge
env:env009","convergence-terraform-env:env013","convergence-terraform-env:env015","convergence-terraform-env:env016","converge
env:env018","convergence-terraform-env:env019","convergence-terraform-env:env020","convergence-terraform-env:env021","converge
env:env023","convergence-terraform-env:env024","convergence-terraform-env:env025","convergence-terraform-env:env026","converge
env:env028","convergence-terraform-env:env029","convergence-terraform-env:env030","convergence-terraform-env:env031","converge
env:env033","convergence-terraform-env:fail","convergence-terraform-env:fail1","convergence-terraform-env:fail10","convergence
env:fail11","convergence-terraform-env:fail13","convergence-terraform-env:fail14","convergence-terraform-env:fail2","converge
env:fail4","convergence-terraform-env:fail5","convergence-terraform-env:fail7","convergence-terraform-env:frimorn","convergen
env:junktest12","convergence-terraform-env:k2_sql","convergence-terraform-env:knees","convergence-terraform-env:licensetest1"
terraform-env:licensetest3","convergence-terraform-env:licensetest4","convergence-terraform-env:licensetest5","convergence-te
env:march26AM","convergence-terraform-env:march26pm","convergence-terraform-env:mattSql","convergence-terraform-env:mattisgre
terraform-env:mh1","convergence-terraform-env:mh2","convergence-terraform-env:mh3","convergence-terraform-env:mh4","convergen
env:mh5","convergence-terraform-env:mh6","convergence-terraform-env:mhdev","convergence-terraform-env:mhdevsql","convergence-
env:mhsql2","convergence-terraform-env:mhsql3","convergence-terraform-env:mhsql4","convergence-terraform-env:monafternoon","c
env:morningtest","convergence-terraform-env:prod","convergence-terraform-env:prod001","convergence-terraform-env:prod002","co
env:prod004","convergence-terraform-env:prod005","convergence-terraform-env:prod010","convergence-terraform-env:prod011","con
env:proxyburn","convergence-terraform-env:proxytest17","convergence-terraform-env:proxytest18","convergence-terraform-env:pro
env:proxytest20","convergence-terraform-env:proxytest742","convergence-terraform-env:proxytest743","convergence-terraform-env
terraform-env:roll3","convergence-terraform-env:roll4","convergence-terraform-env:roll5","convergence-terraform-env:roll6","c
env:roll9","convergence-terraform-env:rolling8","convergence-terraform-env:rollingtest2","convergence-terraform-env:scgwebtes
terraform-env:stage002","convergence-terraform-env:stage003","convergence-terraform-env:stage004","convergence-terraform-env:
terraform-env:stage007","convergence-terraform-env:stage008","convergence-terraform-env:stage009","convergence-terraform-env:
terraform-env:stage012","convergence-terraform-env:stage013","convergence-terraform-env:stage014","convergence-terraform-env:
terraform-env:stage018","convergence-terraform-env:stage020","convergence-terraform-env:stage021","convergence-terraform-env:
terraform-env:stage031","convergence-terraform-env:stage035","convergence-terraform-env:stage036","convergence-terraform-env:
terraform-env:stage039","convergence-terraform-env:stage040","convergence-terraform-env:stage123","convergence-terraform-env:
terraform-env:staging","convergence-terraform-env:stayjing","convergence-terraform-env:stoperroring","convergence-terraform-e
terraform-env:tueafternoon","convergence-terraform-env:tueafternoon2","convergence-terraform-env:tuemorntest","convergence-te
```


Basic Diagram



- Single point of entry for all traffic; NAT gateway for necessary outbound requests
- Separate tiers (Web, App, Data, Domain)
- Load Balancers enable seamless scaling and increase security
- Kubernetes provides scaling and recovery capabilities for platform services
- Can be deployed in multiple AWS regions

The DevOps Pipeline: Jenkins

Jenkins is an open source automation server written in java. Jenkins helps to automate the non-human part of the software development process, with continuous integration and facilitating technical aspects of continuous delivery.

Declarative Pipeline Advantages:

- Provides richer syntactical features over Scripted Pipeline syntax.
- Designed to make writing and reading Pipeline code easier.

The DevOps Pipeline: Jenkins

```
// @Library('platform-helpers@feature/AddEFSProvisionertoClusterDeploy')

agentName = "linux"
agentLabel = "${-> println 'Right Now the Agent Name is ' + agentName; return agentName}"

pipeline {

    agent {label agentName}

    parameters {
        string(name: 'name', description: 'Select the Name for the cluster, you must use a fully qualified domain name (i.e. toolin
        choice(name: 'environment', description: 'The target AWS VPC to deploy the Kubernetes cluster into (required)', choices:'De
        string(name: 'infrastructure_branch', defaultValue: 'develop', description: 'branch name do EFS Provisioner deploy from')
        choice(name: 'kubernetes_version', description: 'Select release of Kubernetes to deploy.', choices:'1.9.3\n1.9.4')
        string(name: 'node_count', description: 'Provide the number of nodes (1-100)', defaultValue: '2')
        string(name: 'master_count', description: 'Provide the number of masters (1-4)', defaultValue: '1')
        choice(name: 'dns_zone', description: 'Select the Hosted DNS Zone for the cluster. (supplychainsoft.com is Dev, supplychain
        choice(name: 'instance_size', description: 'Select VM size for the Instance size for Master and Node(s) (Dev/Prod).', choic
        booleanParam(name: 'protect', description: 'Mark cluster as "Protected" to avoid deletion via Jenkins.', defaultValue: fals
    }


    stages {
        stage('Gathering information for Kubernetes cluster creation...') {

            steps {
                dir('platform') {
                script {
                    echo "Determining correct cluster configuration file and values from Environment parameter..."

                    def kopsConfig = params.environment

                    switch(kopsConfig){
                        case 'Dev':
                            echo "Found kops_config_dev.yaml, using as cluster config..."
                            kopsConf= 'kops_config_dev.yaml'
                            efssubnetIdAZ1= 'subnet-<redacted>'
                            efssubnetIdAZ2= 'subnet-<redacted>'
```

LLamasoft®

Supply Chain By Design

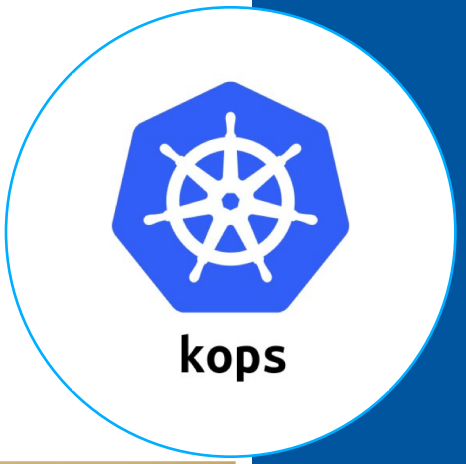
Kubernetes

The DevOps Pipeline: kops

KOPS is an official Kubernetes project for managing production-grade Kubernetes clusters. Kops is currently the best tool to deploy Kubernetes to Amazon Web Services.

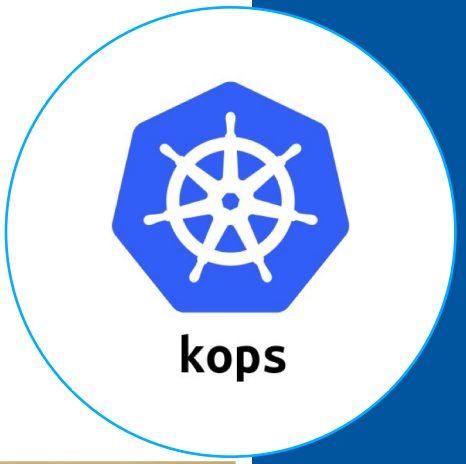
Key Features

- Rolling cluster updates
- Deploy clusters to existing virtual private clouds (VPC) or create a new VPC from scratch
- Provisions single or multiple master clusters



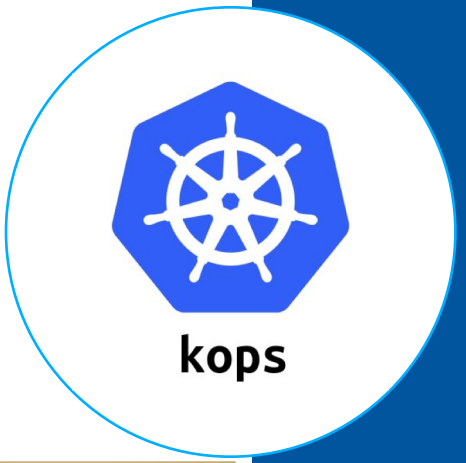
The DevOps Pipeline: kops

```
kops create cluster \
    --node-count 3 \
    --zones us-east-1a,us-east-1b \
    --dns-zone <redacted>.com. \
    --node-size m4.large \
    --master-size m4.large \
    --topology private \
    --vpc vpc-<redacted> \
    --network-cidr <redacted> \
    --networking kube-router \
    --cloud-labels "Environment=\"K8sDev\",Service=\"Kubernetes\",Type=\"INFRA\"" \
    --kubernetes-version 1.9.0 \
    --master-security-groups sg-<redacted> \
    --node-security-groups sg-<redacted> \
    dev.<redacted>.com
```



The DevOps Pipeline: kops

```
// @Library('platform-helpers@feature/AddEFSProvisionertoClusterDeploy')

agentName = "linux"
agentLabel = "${-> println 'Right Now the Agent Name is ' + agentName; return agentName}"

pipeline {

    agent {label agentName}

    parameters {
        string(name: 'name', description: 'Select the Name for the cluster, you must use a fully qualified domain name (i.e. toolin
        choice(name: 'environment', description: 'The target AWS VPC to deploy the Kubernetes cluster into (required)', choices:'De
        string(name: 'infrastructure_branch', defaultValue: 'develop', description: 'branch name do EFS Provisioner deploy from')
        choice(name: 'kubernetes_version', description: 'Select release of Kubernetes to deploy.', choices:'1.9.3\n1.9.4')
        string(name: 'node_count', description: 'Provide the number of nodes (1-100)', defaultValue: '2')
        string(name: 'master_count', description: 'Provide the number of masters (1-4)', defaultValue: '1')
        choice(name: 'dns_zone', description: 'Select the Hosted DNS Zone for the cluster. (supplychainsoft.com is Dev, supplychain
        choice(name: 'instance_size', description: 'Select VM size for the Instance size for Master and Node(s) (Dev/Prod).', choic
        booleanParam(name: 'protect', description: 'Mark cluster as "Protected" to avoid deletion via Jenkins.', defaultValue: fals
    }


    stages {
        stage('Gathering information for Kubernetes cluster creation...') {

            steps {
                dir('platform') {
                script {
                    echo "Determining correct cluster configuration file and values from Environment parameter..."

                    def kopsConfig = params.environment

                    switch(kopsConfig){
                        case 'Dev':
                            echo "Found kops_config_dev.yaml, using as cluster config..."
                            kopsConf= 'kops_config_dev.yaml'
                            efssubnetIdAZ1= 'subnet-<redacted>'
                            efssubnetIdAZ2= 'subnet-<redacted>'
```

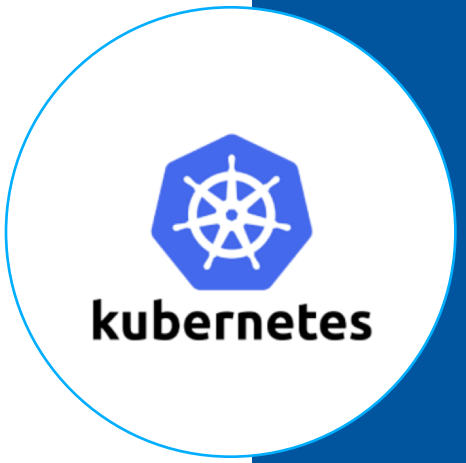


The DevOps Pipeline: Kubernetes

Kubernetes is a portable, extensible open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.

Advantages:

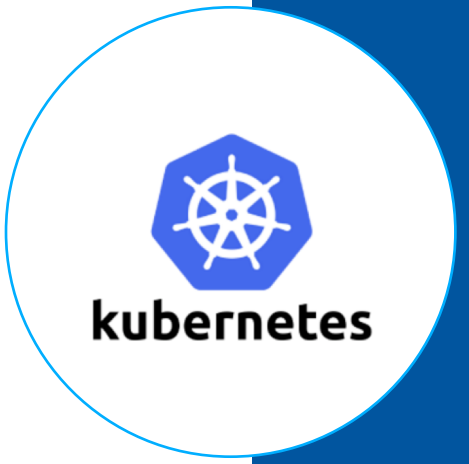
- High scalability
- Easier container management
- Makes rolling-updates *easy*



The DevOps Pipeline: Kubernetes

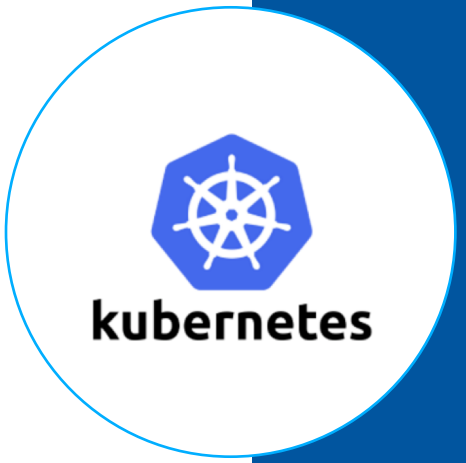
Helpful Hints for Deploying a Cluster:

- Use some sort of Pipeline mechanism for issuing the kops cli / kubectl commands.
- Make the Pipeline tool easy to use by creating a job (Maybe a Jenkins job) that passes parameters into the script you're using.
- Create a docker image with relevant tooling for your team to use. Alpine Linux is a nice lightweight option.



The DevOps Pipeline: Kubernetes

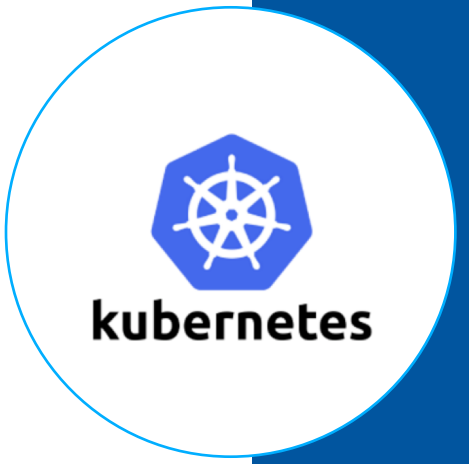
Codeshare



The DevOps Pipeline: Kubernetes

Learning:

- There is a lot of information out there about Kubernetes, some of it is old, some of it is incorrect, and some is from last nights release.
- Do yourself a favor and go take a Pluralsight course on it.
- Additionally, it's good to understand Docker prior to diving into Kubernetes, so maybe take a course on that as well.



Helpful Links:

- Kubectl Cheat Sheet
- Kubernetes Tutorials

The DevOps Pipeline: K8s Lessons Learned

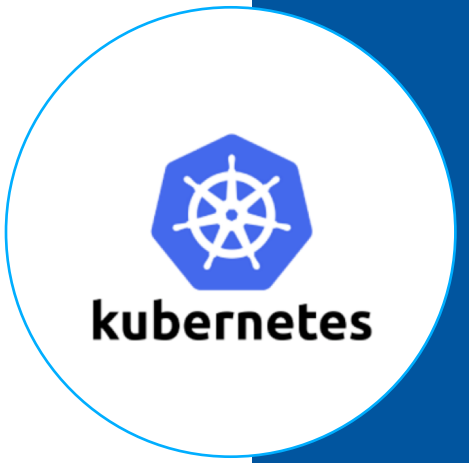
Why isn't my microservice reachable?

Is the service running?
Issue: *kubectl get svc <service name> -n <namespace>*

If the service is running, can I see more detail?
Issue: *kubectl describe svc <service name> -n <namespace>*

What about logs?
Issue: *kubectl get pods –n <namespace> (Get pod name)*
Issue*: kubectl logs pod <podname> -n <namespace>*

Can I tail the logs?
Issue: *kubectl logs –tail=<number of lines> <podname>*

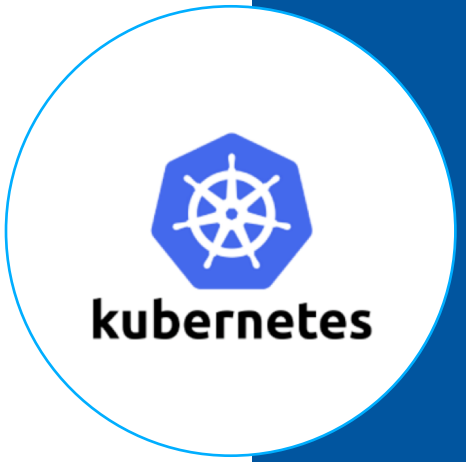


The DevOps Pipeline: K8s Lessons Learned

According to the logs, my microservice is unable to make a connection to the database, what should I do to troubleshoot?

Are the environment variables (connection strings) correct?
Issue: *kubectl get pods -n <namespace> (Get pod name)*
Issue: *kubectl describe pod <podname> -n <namespace>*

Can other microservices connect to the database, if so, can other microservices on the same Node as my microservice connect?
Issue: *kubectl get pods -n <namespace> (Get pod name)*
Issue: *kubectl exec –it <podname> bash/sh -n <namespace>*



LLamasoft®

Supply Chain By Design

What's Next?

The DevOps Pipeline: What's Next?

- Monitoring
  - Prometheus (Grafana, Alerts..etc)
  - Cloudwatch (Infra Alerts)

- Logging
  - Filebeat
  - Fluent-bit/Fluentd

- Tooling
  - Jenkins
  - Artifactory

The DevOps Pipeline: References

- Chef Test Kitchen Tutorial
- Kubectl Cheat Sheet
- Slack Communities
  - Orchestructure
  - Chef
  - Kubernetes

QUESTIONS?

Josh Lynn
email: josh.lynn@llamasoft.com
slack: @Lynny (Made in A2)