Pipelining DevOps with Jenkins and AWS

Jimmy Ray

Me

- DevOps/Cloud Architect

- Early-adopter and fast-follower of Cloud, DevOps, and Java technologies

- "Serial Speaker"

- Wrote Consul-KV-Builder Plugin (wildly-popular? - not so much)
  - https://wiki.jenkins-ci.org/display/JENKINS/Consul-KV-Builder+Plugin

- Blog: http://www.techsand.com

- LinkedIn: http://www.linkedin.com/in/iamjimmyray

KEEP
CALM
AND
USE
JENKINS

Disclaimer – "The Buck Stops Here!"

- The ideas presented, and maybe even espoused, today are my own thoughts, and do not reflect or represent those of my employer, customers, or colleagues.

Agenda

- Today's Architecture
  - EC2 (Roles, Connections, etc.)
  - Today's Jenkins
  - Today's Pipeline

- AWS CodeCommit

- AWS CodeDeploy

- AWS Pipeline
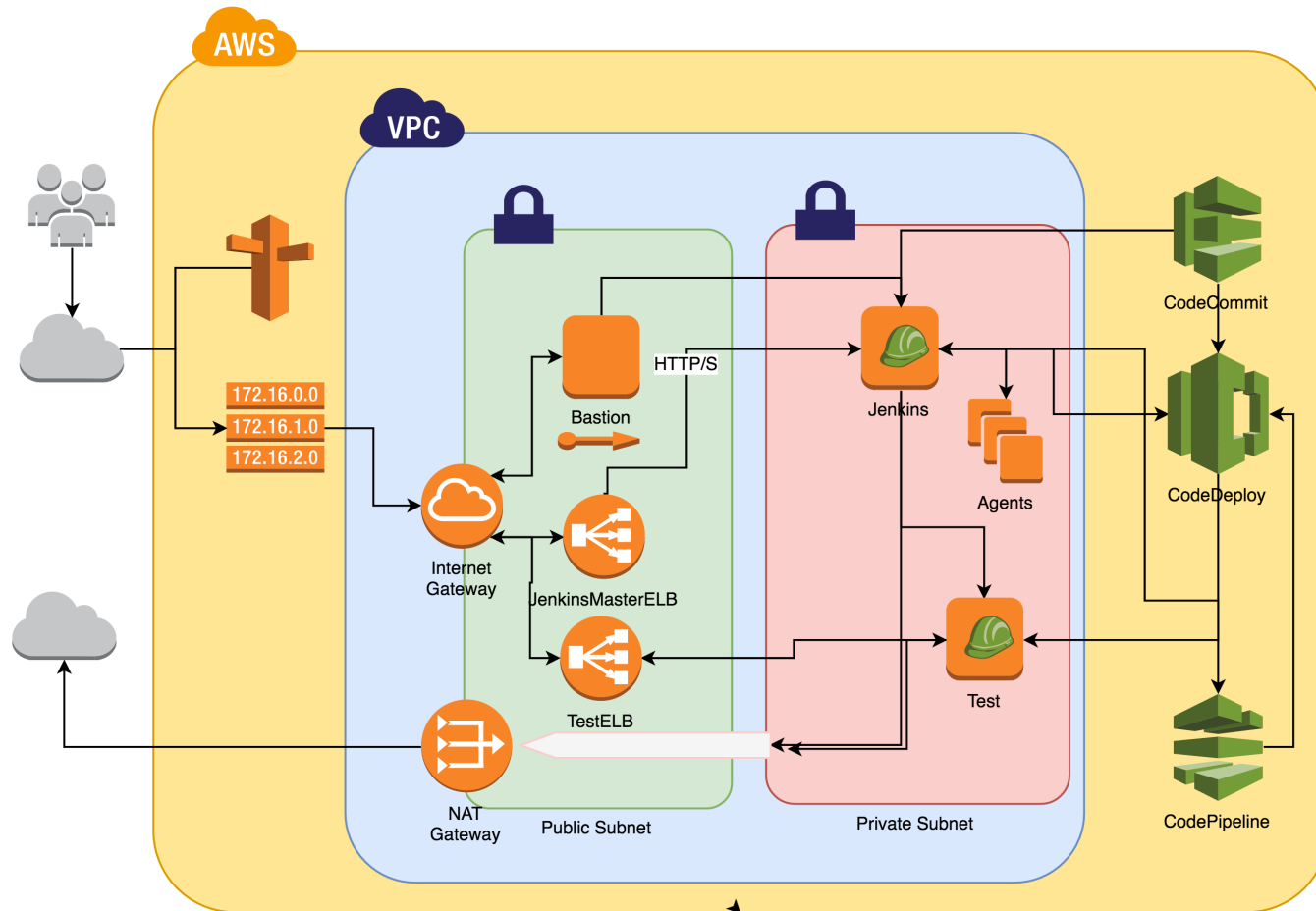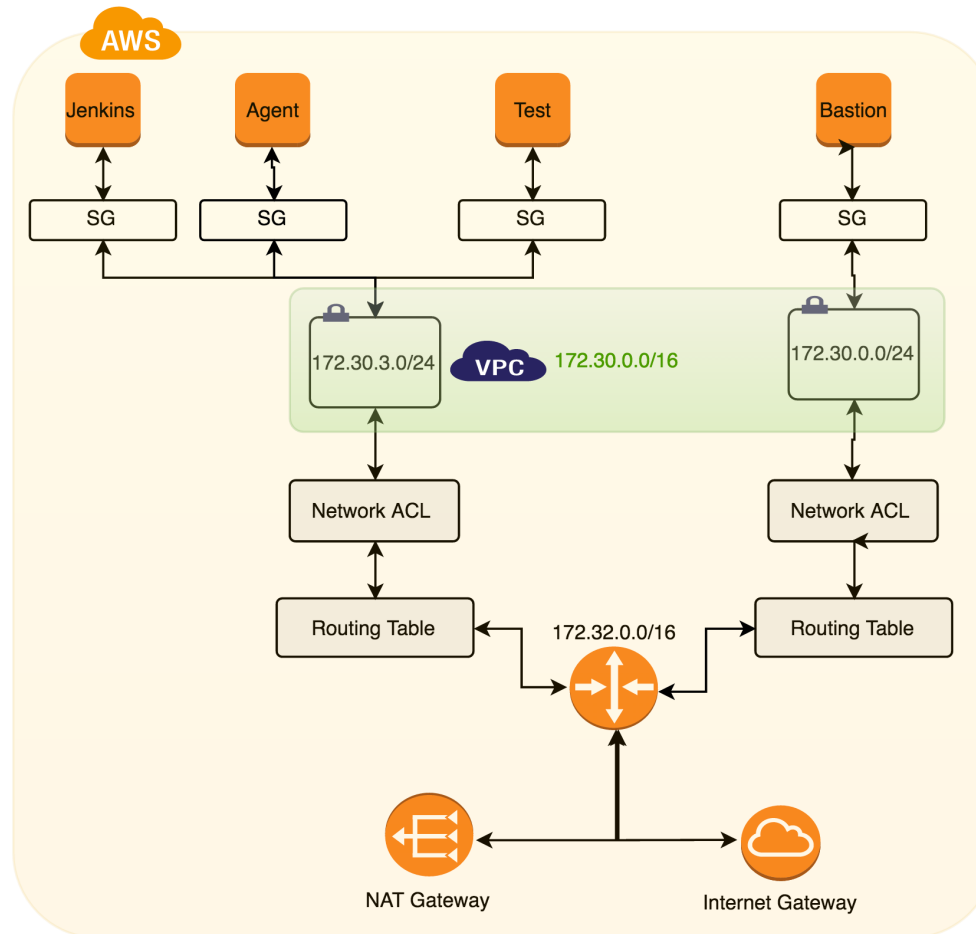
- Best Practices

- Q and (maybe) A

Architecture

AWS

VPC

172.16.0.0
172.16.1.0
172.16.2.0

Internet Gateway

Bastion

JenkinsMasterELB

TestELB

NAT Gateway

Public Subnet

HTTP/S

Jenkins

Agents

Test

Private Subnet

CodeCommit

CodeDeploy

CodePipeline

#JenkinsWorld

Architecture

AWS

Jenkins	Agent	Test	Bastion
SG	SG	SG	SG

172.30.3.0/24   VPC   172.30.0.0/16   172.30.0.0/24

Network ACL   Network ACL

Routing Table   172.32.0.0/16   Routing Table

NAT Gateway   Internet Gateway

#JenkinsWorld

SSH Agent Forwarding with Bastion Host

- Access Bastion host (with EIP) in public subnet using PEM key file and SSH.
  - Will access others server via the Bastion host.

- Use SSH forwarding so that PEM file does not need to be stored on Bastion host

- Good solution for keeping hosts in private subnets without Internet Gateway (IGW)
  - Use route to NAT Gateway in public subnets

EC2 Roles for Your Jenkins Instances

- EC2 instances can be assigned IAM roles
  - Must be assigned when EC2 is launched.
- These roles can then be used by processes on the EC2, including Jenkins

EC2 Instance Metadata

- Data about your EC2 instance that you can use from within your EC2 instance.
- http://169.254.169.254/latest/meta-data/iam/security-credentials/JenkinsWorld2016JenkinsMaster

Jenkins Tools

- Java 8
- NGINX
- Git
- AWS CLI
- Maven 3.3.9
- jq (https://stedolan.github.io/jq/)

Choose Your Jenkins Distro Wisely

- I like to use the **Jenkins Long-Term Support release**
  - https://wiki.jenkins-ci.org/display/JENKINS/LTS+Release+Line

sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/**redhat-stable**/jenkins.repo

sudo rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key

sudo yum install jenkins

sudo service jenkins start/stop/restart

sudo chkconfig jenkins on

curl localhost:8080

Choose Your Jenkins Distro Wisely (continued)

- I like the LTS versions, but I wanted/needed the latest features

sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/**redhat**/jenkins.repo

sudo rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key

sudo yum install jenkins

sudo service jenkins start/stop/restart

sudo chkconfig jenkins on

curl localhost:8080

- New syntax for stage definition:
  - "Using the 'stage' step without a block argument is deprecated"

```
//Checkout
def checkoutCode() {
    stage ('Checkout') {
        git url: GIT_URL
    }
}
```

New Shell Step Syntax

- Advanced configuration added
  - Now can return the standard output to variable assignment
  - No longer need "sentinel" files

```
echo 'Install CodeDeploy Agent'
PROPS_MAP['codeDeployAgentStatus'] = sh (returnStdout: true, script: """
    ssh -o StrictHostKeyChecking=no -i $keyPath $PROPS_MAP.ec2User@$PROPS_MAP.ip 'wget $PROPS_MAP.codeDeployInstallBucket
    && chmod +x ./install && sudo ./install auto && sudo service codedeploy-agent status'
""")
```
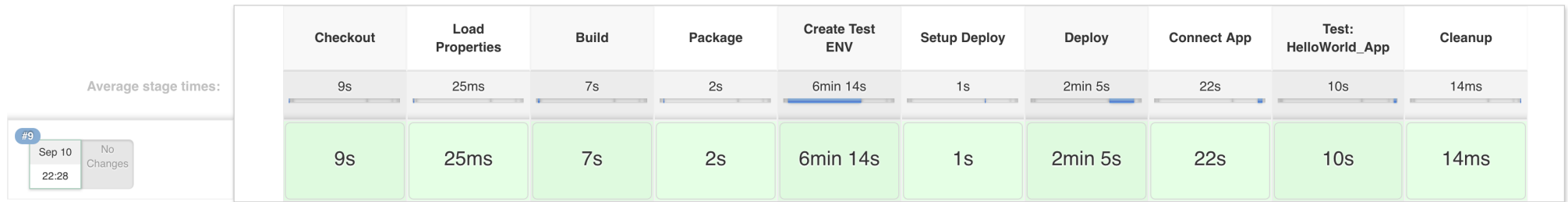
Today's Jenkins Pipeline

CommitDeployPipeline - Stage View

	Checkout	Load Properties	Build	Package	Create Test ENV	Setup Deploy	Deploy	Connect App	Test: HelloWorld_App	Cleanup
Average stage times:	9s	25ms	7s	2s	6min 14s	1s	2min 5s	22s	10s	14ms
#9 Sep 10 22:28 — No Changes	9s	25ms	7s	2s	6min 14s	1s	2min 5s	22s	10s	14ms

Pipeline Utilities

- Use pipeline-utility-steps plugin
  - https://wiki.jenkins-ci.org/display/JENKINS/Pipeline+Utility+Steps+Plugin
  - https://jenkins.io/doc/pipeline/steps/pipeline-utility-steps/

- Setup tooling (Maven, Sonar, etc.) and reference this tooling in CPS

```
node ('master') {
    stage 'Build'
    git 'https://git-codecommit.us-east-1.amazonaws.com/v1/repos/JavaOne2014'
    def mvnHome = tool 'Maven339'
    def pom = readMavenPom file: 'pom.xml'
    def version = pom.version.replace("-SNAPSHOT", ".${currentBuild.number}")
    sh "${mvnHome}/bin/mvn clean package"
```

- Watch your environment variables
  - If used, put exports in the same shell call as AWS CLI call (context)
- Use the CloudBees Enterprise Jenkins AWS CLI Plugin
  - Or, create IAM user and use access key and secret keys, with **withEnv** construct
  - Or, use –-profile switch in AWS CLI call
  - Or, use EC2 roles and credentials from EC2 metadata
  - Or, set Jenkins AWS system-level configuration

CloudBees AWS CLI Plugin

- https://go.cloudbees.com/docs/cloudbees-documentation/cje-user-guide/chapter-aws-cli.html
- Uses pipeline build wrapper to wrap AWS CLI calls in environment with proper credentials.
- Allows you to manage AWS CLI credentials from within Jenkins

CloudBees AWS CLI Plugin

```
Script    1 ▾ node ('master') {
          2 ▾     wrap([$class: 'AmazonAwsCliBuildWrapper',
          3               credentialsId: 'jw2016',
          4 ▾             defaultRegion: 'us-east-1']) {
          5
          6               sh '''
          7                  aws ec2 describe-instances
          8               '''
          9         }
         10 }
```

cd /var/lib/jenkins

sudo -u jenkins aws configure

AWS CLI - Create Test ENV Stage

- Create EC2 instance from known AMI, Security group, Subnet, Key Pair, Role
  - Write JSON results to file
  - Slurp file to get InstanceId, PrivateIpAddress
- Wait for instance to be "running"
  - Need InstanceId to call "wait"
- Tag instance
  - Need InstanceId
- Wait for instance to be reachable
  - Then configure instance using ssh and PrivateIpAddress

Jenkins World
2016

- Install tools (Java, Tomcat, etc.)
- Install AWS CodeDeploy agent

jq

- Used in pipeline scripts to parse JSON using JMESPath
- Use JMESPath (JSON Matching Expression path) Terminal (jpterm) to build paths
  - https://github.com/jmespath/jmespath.terminal

```
sudo su –
cd /usr/bin
wget http://stedolan.github.io/jq/download/linux64/jq
chmod +x ./jq
```

AWS CodeCommit

- Git repos in AWS
- Requires credential helper in .gitconfig

git config --global credential.helper '!aws --profile JenkinsWorld2016 codecommit credential-helper $@'

git config --global credential.UseHttpPath true

```
[credential]
    helper = !aws --profile JenkinsWorld2016 codecommit credential-helper $@
    UseHttpPath = true
```

AWS CodeCommit – AWS Temp Passwords

- If you are using OSX with CodeCommit you need to be aware of the issue caused by AWS and your OSX Keychain utility
  - AWS uses a temporary that will cause a 403 error if your OSX Keychain is not configured properly
  - http://docs.aws.amazon.com/codecommit/latest/userguide/setting-up-https-unixes.html#setting-up-https-unixes-credential-helper
  - Temp Fix: `security delete-internet-password -l git-codecommit.us-east-1.amazonaws.com`

AWS CodeCommit Repos

Dashboard

Share and manage your code in the cloud with AWS CodeCommit. Create, edit, and view details about your code repositories.

Create new repository

▼ Filter by repository name		« ‹ 1 to 4 of 4 Repositories ›

Name ▼	Description	L
JavaOne2014	Demo app	1
jenkins-master-non-lts	Config changes for Jenkins Master (non-LTS)	1
jenkins-master	Config changes for Jenkins Master	S
pipeline-global	Global scripts for Jenkins Pipeline jobs.	S

« ‹ 1 to 4 of 4 Repositories ›

AWS CodeDeploy

- CodeDeploy is used to setup deployment resource targets, groups, and deployment configurations.

- Deploys to EC2 or ASG
  – Allows for rollback via redeploy

- CodeDeploy Agent must be running on targets to receive deployments
  – Uses appspec.yml to customize deployment when the deployment package is delivered to the instance.

```
[ec2-user@ip-172-32-1-8 ~]$ sudo service codedeploy-agent status
The AWS CodeDeploy agent is running as PID 6914
[ec2-user@ip-172-32-1-8 ~]$ 
```
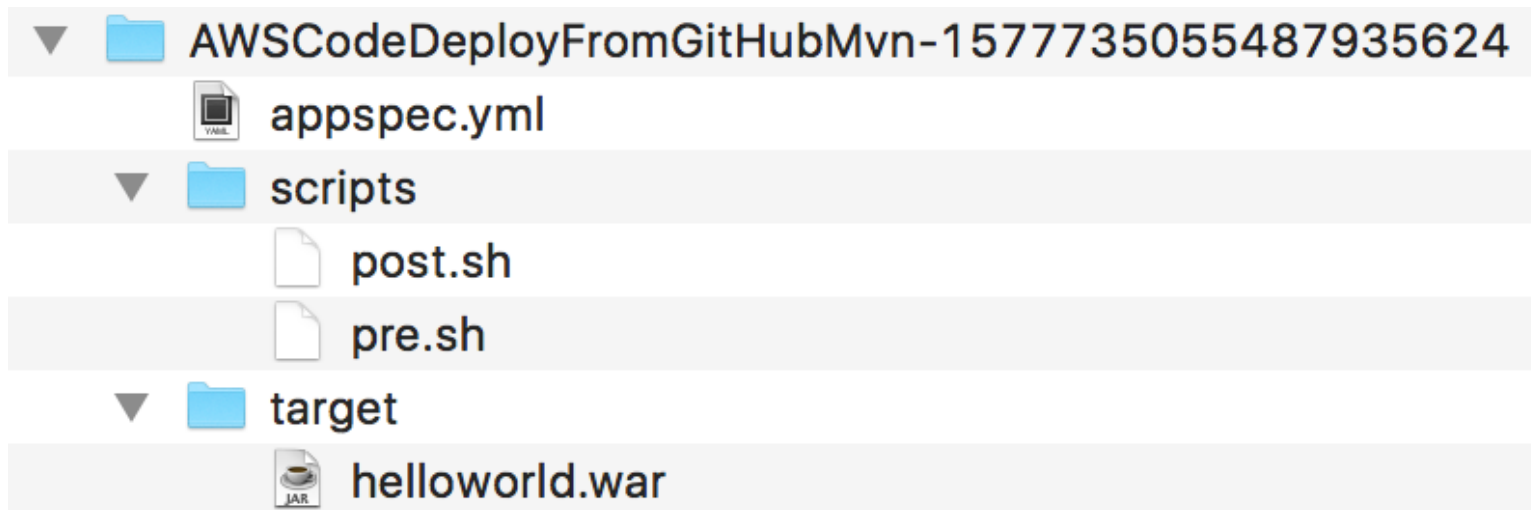
CodeDeploy Packaging

- Package artifacts to deploy via the CodeDeploy
  - This is a ZIP file with the artifacts as well as the YAML (*.yml) file needed for CodeDeploy automation

▼ 📁 AWSCodeDeployFromGitHubMvn-1577735055487935624

    📄 appspec.yml

    ▼ 📁 scripts

        📄 post.sh

        📄 pre.sh

    ▼ 📁 target

        📄 helloworld.war

AWS CodePipeline

- Uses CodeDeploy to target resources
- Uses the concept of:
  - providers
    - Source, Deployment, Build, etc.
  - Workers
  - Actions
- Can use S3 (GitHub, CodeCommit) as pipeline source
  - S3 bucket versioning is required
  - Lifecycle Management recommended
- Can use Jenkins for build integration
- AWS CLI call to create CodePipeline
  - Requires JSON config file

CodeDeploy and CodePipeline Role Assumptions

```
{
  "Sid": "",
  "Effect": "Allow",
  "Principal": {
    "Service": "codedeploy.amazonaws.com"
  },
  "Action": "sts:AssumeRole"
},
{
  "Sid": "",
  "Effect": "Allow",
  "Principal": {
    "Service": "codepipeline.amazonaws.com"
  },
  "Action": "sts:AssumeRole"
}
```

Jenkins World 2016

#JenkinsWorld

Timestamped Logs

- timestamps { //code block }
- Adds timestamped log entries
- Adds additional UI control when reviewing logs.

Timestamps — View as plain text

- System clock time
  - Use browser timezone
- Elapsed time
- None

Best Practices and Possible Ideas

- Jenkins SCM Sync Configuration Plugin
- Bastion Hosts with SSH Forwarding
  – or centralized security (example: Centrify)
- Groovy Sandbox and Script Approval
  – Look out for serialization issues
  – @NonCPS

Possible Next Steps...

- Move from Bastion host access to Centrify or similar tools.
  - Reduce compromised credentials, stop using key-pairs
- Externalize scripts to shared libraries and Jenkinsfile
  - Keep your code **DRY** (**D**on't **R**epeat **Y**ourself)
- Capture shell outputs to files to reduce noise in logs
- Replace shell-script configurations with CloudFormation or Terraform
- Add Chef or Ansible
- Move from EC2 to ECS (if/when CodeDeploy works with ECS)
- Move from master to Jenkins Agent node

Final Thoughts…

- There is a lot of room for improvement to the integration between Jenkins Pipeline and AWS
  - Plugins to be written and/or modified to be used with AWS resources

- Jenkins Pipeline is the "DevOps Equalizer"
  - Instead of having to memorize the look and feel of many different plugins, CPS syntax and lexicon is required.
  - Programmers who have had a strong desire to use Jenkins are now re-empowered

- I would really like to be able to disable `sandbox=true` for `CpsScmFlowDefinition` jobs.

- I would really like to increase the size of the CodeMirror script editor in the pipeline-job configuration screen

Questions?

#JenkinsWorld

Jenkins World
2016

#JenkinsWorld

Pipeline Shared Libraries

- https://github.com/jenkinsci/workflow-cps-global-lib-plugin/blob/master/README.md

- Helps you keep your pipeline code DRY

- Can use any GitRepo, even CodeCommit
  - https://github.com/cloudbees/jenkins-scripts/blob/master/pipeline-global-lib-init.groovy

Pipelines That Have Lost Their Way

- Occasionally, pipelines stop responding
  - They may even show that they were "aborted"
  - They are still "spinning"

- In the Global Security, disable "Prevent Cross Site Request Forgery exploits"

- Then delete the job
  - $JENKINS_URL/job/${JOB_NAME}/${BUILD_NUMBER}/doDelete
    - o Hit the "Try Posting" button if it appears
    - o May need to disable "Prevent Cross Site Request Forgery exploits" in Global Security.

- You may need to restart Jenkins: $JENKINS_URL/restart

Console Output

```
Started by user Jimmy Ray
[Pipeline] node
Running on master in /var/lib/jenkins/workspace/JenkinsWorld2016/IAmLost
[Pipeline] {
[Pipeline] stage (GetLost)
Entering stage GetLost
Proceeding
[Pipeline] echo
1
Aborted by Jimmy Ray
Click here to forcibly terminate running steps
```