



# Securing your DevOps Toolchain in the Cloud



Cisco Public 2017 All Rights Reserved

Tony Rice  
Cisco Security & Trust Organization

# Prerequisites

If you don't already have these established,  
please take care of this now



Amazon AWS account (free tier is fine)

[aws.amazon.com/free](https://aws.amazon.com/free)



Github.com account

# Python Prep (for local testing)

- Install Python

<https://www.python.org/downloads/>

- Install PIP

<https://pip.pypa.io>

`pip install <module>`

- `virtualenv`

- `pip install flask==0.10.1`

# About Us

Tony Rice

Senior Information Security Engineer

RTP



| Certified Information  
Systems Security Professional

Khadija Amin

Information Security Architect

SJC



SEC-USR-MESS-2 SEC-RUN-SAFEC SEC-RUN-OSC SEC-RUN-ASLR SEC-PWD-MAXLIFE SEC-PWD-AUDIT SEC-OPS-RDY  
SEC-0PS-NPR0D SEC-NRCV-CRED-5 SEC-LOG-TENIS0L-2 SEC-LOG-STATCHG SEC-LOG-LOGCONF SEC-LOG-LIMIT  
SEC-LOG-CONTENT SEC-LOG-ATTACK SEC-LOG-ADMIN SEC-IPS-ESP-5 SEC-INSEC-USER SEC-CSP-LOGTRUST SEC-  
CRY-LOG SEC-AUT-LOGZCHG SEC-AUT-LOGNCHG SEC-WEB-HTTPMETH-2 SEC-SRT-311000-3 SEC-OPS-BUSCONT-2 S  
EC-LOG-CENTRAL-3 SEC-INF-HARDEN-2 SEC-509-FQDN-2 SEC-SET-TIME SEC-SRT-311000-2 SEC-LOG-INDC-2 S  
EC-SRT-DTLS-2 SEC-CHG-LOGD-3 SEC-CRE-NOLOCK SEC-IP-IPv6 SEC-AUD-FIELD-3 SEC-OPS-BUSCONT SEC-INF-  
-DOS SEC-INF-AVSCAN SEC-OPS-REVOKE SEC-DAT-RETAIN SEC-DAT-REMOVE SEC-DAT-AUDIT SEC-WEB-NOREDIR  
SEC-WEB-IDCACHE SEC-WEB-IDCTX SEC-SDP-311030 SEC-SRT-311020 SEC-SRT-311010 SEC-TIME-NTP SEC-ASU-  
-TRAIN-2 SEC-WEB-SMTIME-2 SEC-WEB-STATE SEC-WEB-CRYPTOCOOK-2 SEC-WEB-RESP-2 SEC-WEB-SESCOOK-2 SEC-  
-OPS-MAXLIFE SEC-DAT-VMCRYPT SEC-WEB-IDATTR-2 SEC-AUT-CREROT SEC-LOG-CENTRAL-2 SEC-AUT-LOG SEC-  
-LOG-RET SEC-WEB-NPCOOKIE-2 SEC-DAT-SANITIZE SEC-DAT-KEYMGMT SEC-ALL-LOGD-2 SEC-SW-NEEDONLY SEC-  
-ASU-STATIC SEC-HTP-SSL3-3 SEC-ASU-TMOD SEC-OPS-LEGAL SEC-OPS-INCPOL SEC-INF-MGTZONE SEC-CRE-PRI  
V SEC-CRE-MULTIFAC SEC-DAT-GEOG SEC-DAT-SEGR SEC-DAT-BACKUP SEC-DAT-MEDIA SEC-WEB-CLKJACK SEC-  
-WEB-IDVALID SEC-PRV-KNOWWHAT SEC-PRV-NOCOLL SEC-PRV-USERAUTH SEC-PRV-ERASE SEC-DAT-PRIV-2 SEC-P  
RV-LOCALID SEC-PRV-MANAGE SEC-TLS-CURR-3 SEC-LOG-FORM SEC-OPS-RESTCRYPT-2 SEC-WEB-XSSUBCXT-2 SEC-  
-WEB-HTTPMETH SEC-WEB-SQLIN-2 SEC-509-CHAIN SEC-DAT-BACKUPTS SEC-LOG-TRANS SEC-DAT-SOP SEC-WEB-  
-URLPARAM SEC-WEB-SYSIN-2 SEC-WEB-ACCBYURL-2 SEC-LOG-PROTO SEC-LOG-APPREQ SEC-WEB-IDALL-2 SEC-LOG-  
-ACCESS-2 SEC-WEB-CSRF SEC-DAT-BACKUPER SEC-DAT-BACKUPI SEC-509-LIFETIME SEC-509-CERTEXT SEC-CR  
E-LIMTRY-2 SEC-LOG-NOSENS-2 SEC-WEB-INJIN-2 SEC-WEB-SANIHTML SEC-WEB-AUTOCOMP SEC-DAT-BACKUPET  
SEC-LOG-INFRAREQ SEC-WEB-ID-2 SEC-DAT-ISMS-2 SEC-WEB-XSS SEC-509-FQDN SEC-HTP-HSTS SEC-DAT-MONE  
Y SEC-WEB-XPATHIN-2 SEC-509-REVOKE SEC-SUP-PATCH SEC-UPS-REGI SEC-INT-CRED-2 SEC-DEF-CRED-2 SEC-  
-OPS-CHGMGT SEC-WEB-SCAN SEC-INF-SCAN SEC-INF-THIRD SEC-INF-DEVNET SEC-INF-ZONES SEC-INF-MONITO  
R SEC-INF-FWSEP SEC-INF-VER SEC-INF-HARDEN SEC-OPS-ASTMGT-2 SEC-OUT-CRED-2 SEC-OPS-STRENGTH SEC-  
-CRY-PRIM SEC-OPS-SEG-2 SEC-AUT-ACCDEF SEC-CRY-STDCODE SEC-CRE-SHARE SEC-OPS-SUDO SEC-AUT-AUTH  
SEC-OPS-PUBCRYPT-2

# In this workshop we will setup

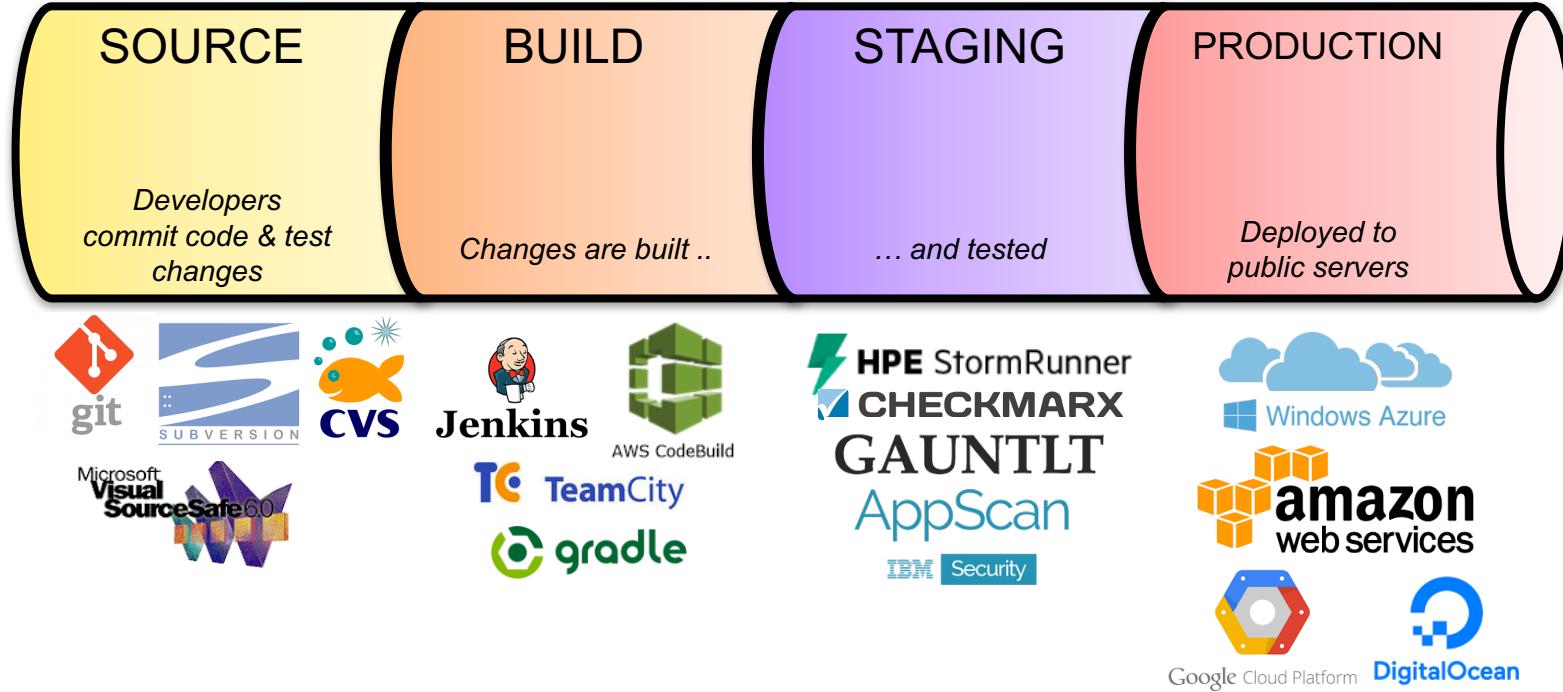
- A source code repository
- A centralized build server
- Deploy a simple micro service into the cloud



Along the way, we'll form them into a continuous integration toolchain ... securely



# Code pipeline



# Our Continuous Delivery Pipeline

Source Code Management

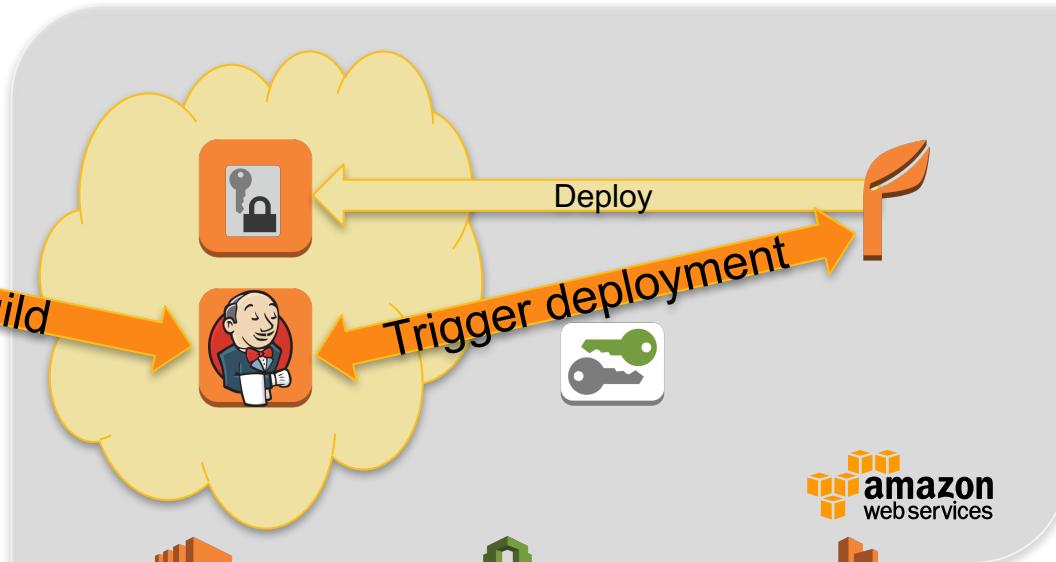


GitHub

Build

Identity

Deployment



EC2



IAM



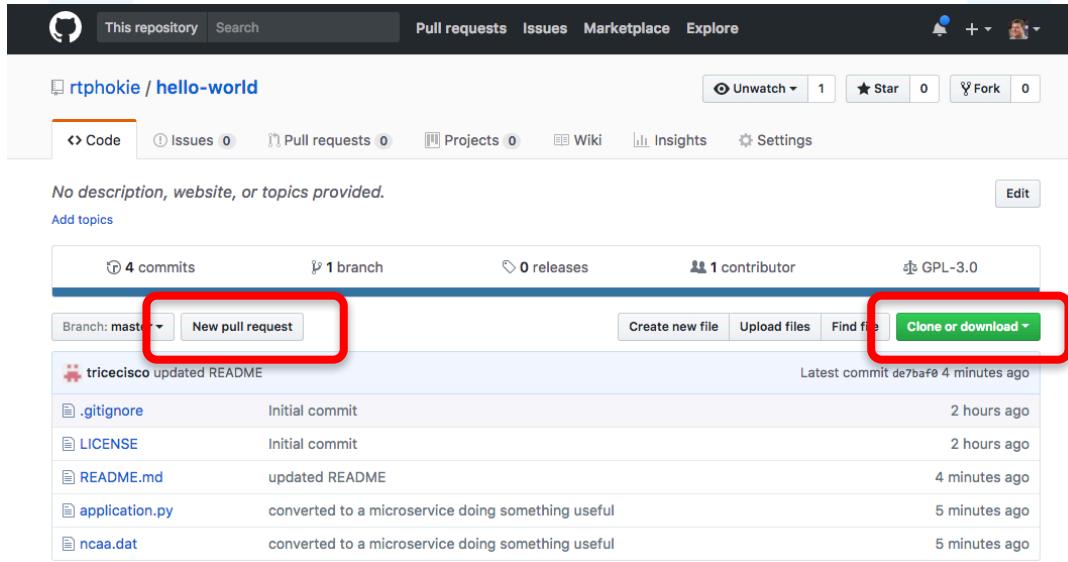
Elastic  
Beanstalk

# Agenda

- 8:30-9:30 CI/CD overview / AWS and Git prerequisites
- 9:30-10:00 AWS Overview / Fork the GitHub Hello Word Repo
- 10:00:10:30 Provision EC2, Elastic Beanstalk and security groups
- 10:30-10:45 Break
- 10:45 - noon install Jenkins with Git and EB plugins
- break
- Noon-1:30 lunch
- 1:30 – 2:00 Vault overview
- 2-3:00 vault install and config
- 3-4 instrument Jenkins with vault secrets
- 4-5 build and deploy Hello World
- 5-5:30 wrap up and best practices

# Get the app source code from GitHub

- Visit <https://github.com/rtpphokie/hello-world>



Pull ...  
then clone

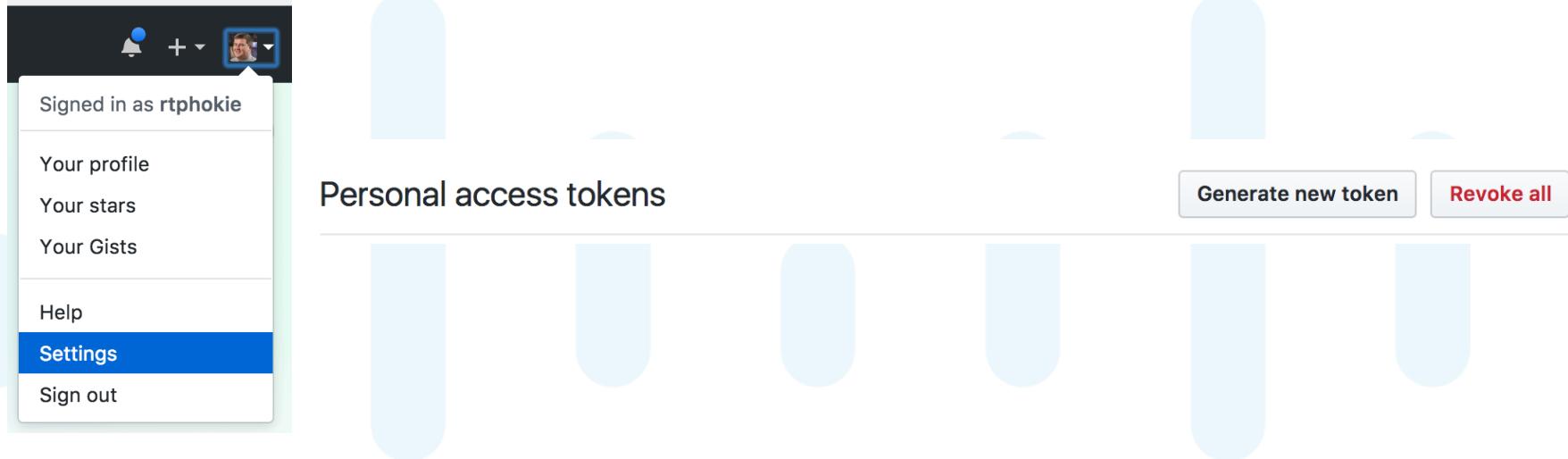
Clone

# Join seccon\_bootcamp organization

- [https://github.com/seccon\\_bootcamp](https://github.com/seccon_bootcamp)
- Or look for an invite in email

# Create a GitHub developer key

Settings -> Developer Settings -> Personal access tokens



# Get the app running locally

1. virtualenv ~/eb-virt
2. source ~/eb-virt/bin/activate
3. git https://github.com/rtpfokke/hello-world.git  
or download from Box (<http://bit.ly/isaca-hello-world>)
4. cd hello-world
5. pip install -r requirements

optional

# Test application locally

1. `python application.py`
2. Visit `http://localhost:5000`

Hello NCAA

This is a RESTful micro service returning information on NCAA Div I football teams

- *param/search string*  
where *param* is school, conference, mascot, city or state and

example: `http://localhost:5000/school/bama`

optional

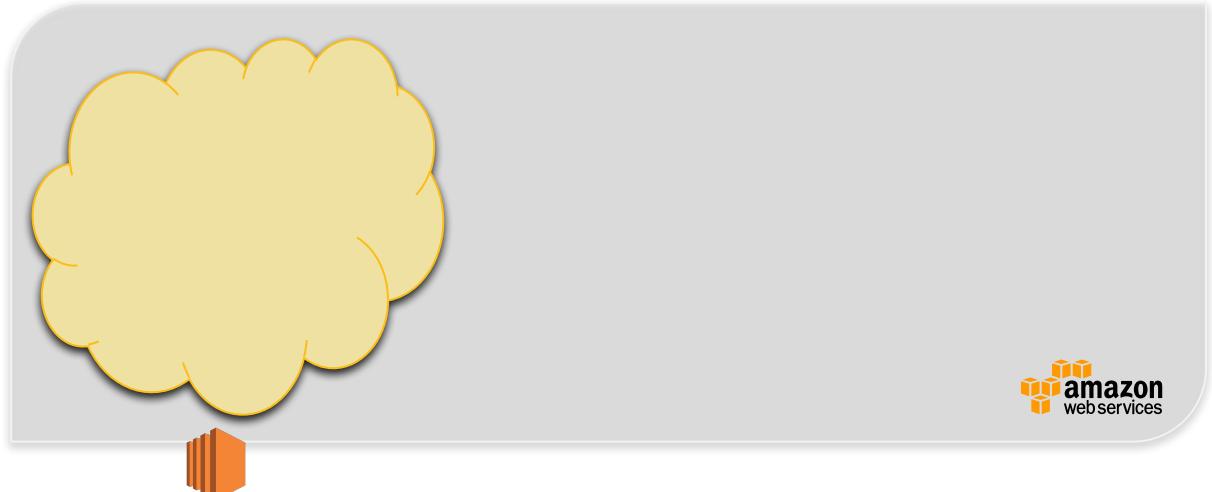


# Our Continuous Delivery Pipeline

Source Code  
Management



GitHub



EC2



# What is Vault?

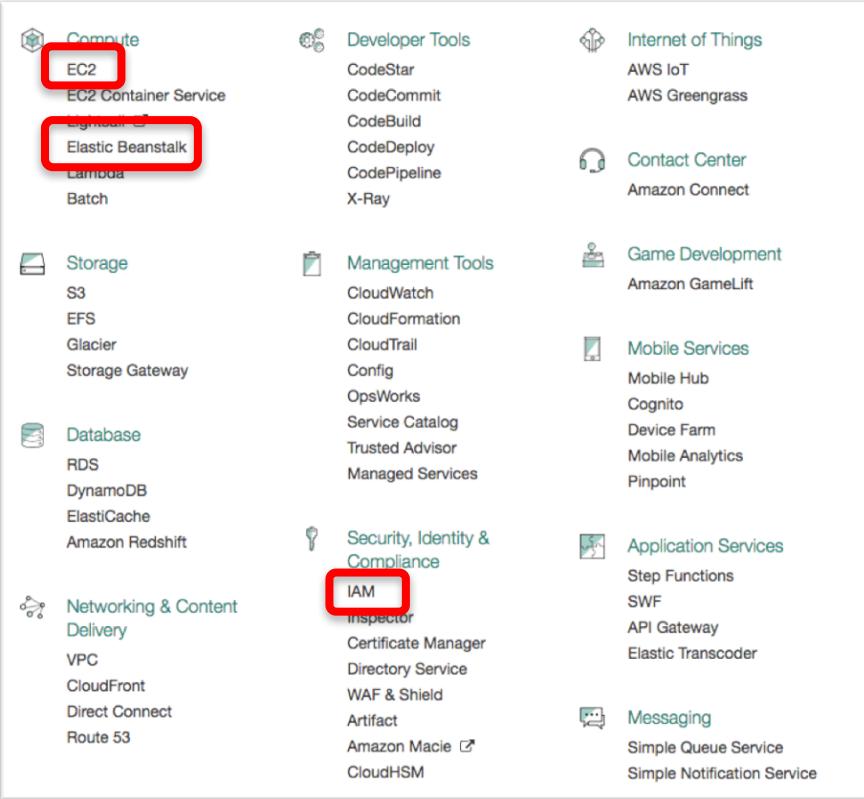
- A tool for Managing secrets.
- Software KMS (key management system).
- **Vault** secures, stores, and tightly controls access to secrets.



# Agenda

- 8:30-9:30 CI/CD overview / AWS and Git prerequisites
- 9:30-10:00 AWS Overview / Fork the GitHub Hello Word Repo
- 10:00:10:30 Provision EC2, Elastic Beanstalk and security groups
- 10:30-10:45 Break
- 10:45 - noon install Jenkins with Git and EB plugins
- break
- Noon-1:30 lunch
- 1:30 – 2:00 Vault overview
- 2-3:00 vault install and config
- 3-4 instrument Jenkins with vault secrets
- 4-5 build and deploy Hello World
- 5-5:30 wrap up and best practices

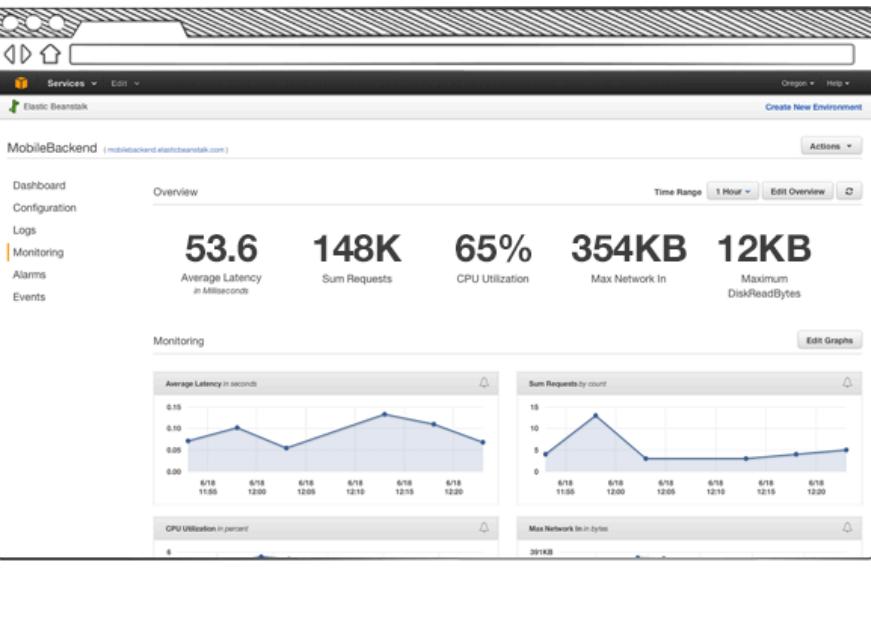
# AWS console



We will be using

- EC2
  - Security Groups
  - Instances
- IAM
- Elastic Beanstalk

# Elastic Beanstalk: Provision



## Welcome to AWS Elastic Beanstalk

With Elastic Beanstalk, you can **deploy**, **monitor**, and **scale** an application quickly and easily. Let us do the heavy lifting so you can focus on your business.

To deploy your **existing web application**, create an [application source bundle](#) and then [create a new application](#). If you're using **Git** and would prefer to use it with our command line tool, please see [Getting Started with the EB CLI](#).

To deploy a **sample application** with just one click, select a platform and click **Launch Now**.

By launching the sample application, you allow AWS Elastic Beanstalk to administer AWS resources and necessary permissions on your behalf. [Learn more](#)

Python

Looking for a different platform? [Let us know.](#)

AWS Elastic Beanstalk will create an environment running Python 3.4 on 64bit Amazon Linux 2017.03 v2.5.2. [Change platform version](#)

**Launch Now**

2.7 on 64 Bit Linux



# Elastic Beanstalk provision

- Create Environment
  - Create Webserver (Python, change to 2.7; single instance)
  - Use sample application
  - Launch

Environment Information

Enter your environment information.

Environment name:

Environment URL:   Optional: 200 character maximum

Description:

# IAM: Provision EB User

The screenshot shows the AWS IAM 'Add user' wizard and the 'Set permissions' screen for a user named 'Jenkins2EB'.

**Add user Wizard:**

- Step 1: Set user details**
  - User name: Jenkins2EB
  - Access type: Programmatic access (selected)
- Step 2: Set permissions**
  - Policy: AWSElasticBeanstalkFullAccess (selected)

**Set permissions for Jenkins2EB:**

- Permissions:** Add user to group, Copy permissions from existing user, Attach existing policies directly.
- Policies:** Filtered by 'Policy type' to show 'AWS managed'. One policy is listed:
  - Policy name: AWSElasticBeanstalkFullAccess
  - Type: AWS managed
- Buttons:** Previous, Next: Review (highlighted with a red box), Create user (highlighted with a red box).

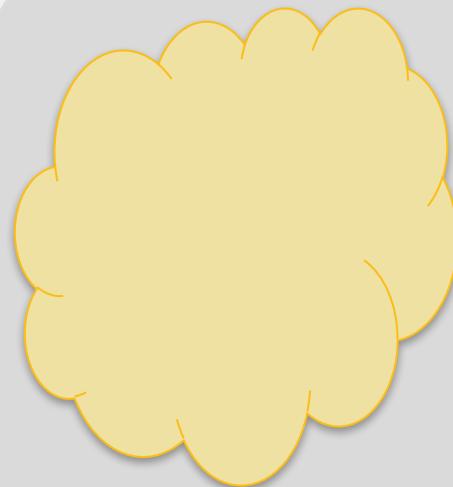
# Our Continuous Delivery Pipeline

Source Code  
Management



GitHub

Build



EC2

Identity



IAM

Deployment



Elastic  
Beanstalk

# Agenda

- 8:30-9:30 CI/CD overview / AWS and Git prerequisites
- 9:30-10:00 AWS Overview / Fork the GitHub Hello Word Repo
- 10:00:10:30 Provision EC2, Elastic Beanstalk and security groups
- 10:30-10:45 Break
- 10:45 - noon install Jenkins with Git and EB plugins
- break
- Noon-1:30 lunch
- 1:30 – 2:00 Vault overview
- 2-3:00 vault install and config
- 3-4 instrument Jenkins with vault secrets
- 4-5 build and deploy Hello World
- 5-5:30 wrap up and best practices

# EC2: security group

Create Security Group

Security group name: Jenkins 8080

Description: Jenkins 8080

VPC: vpc-4ac6d728 (default)

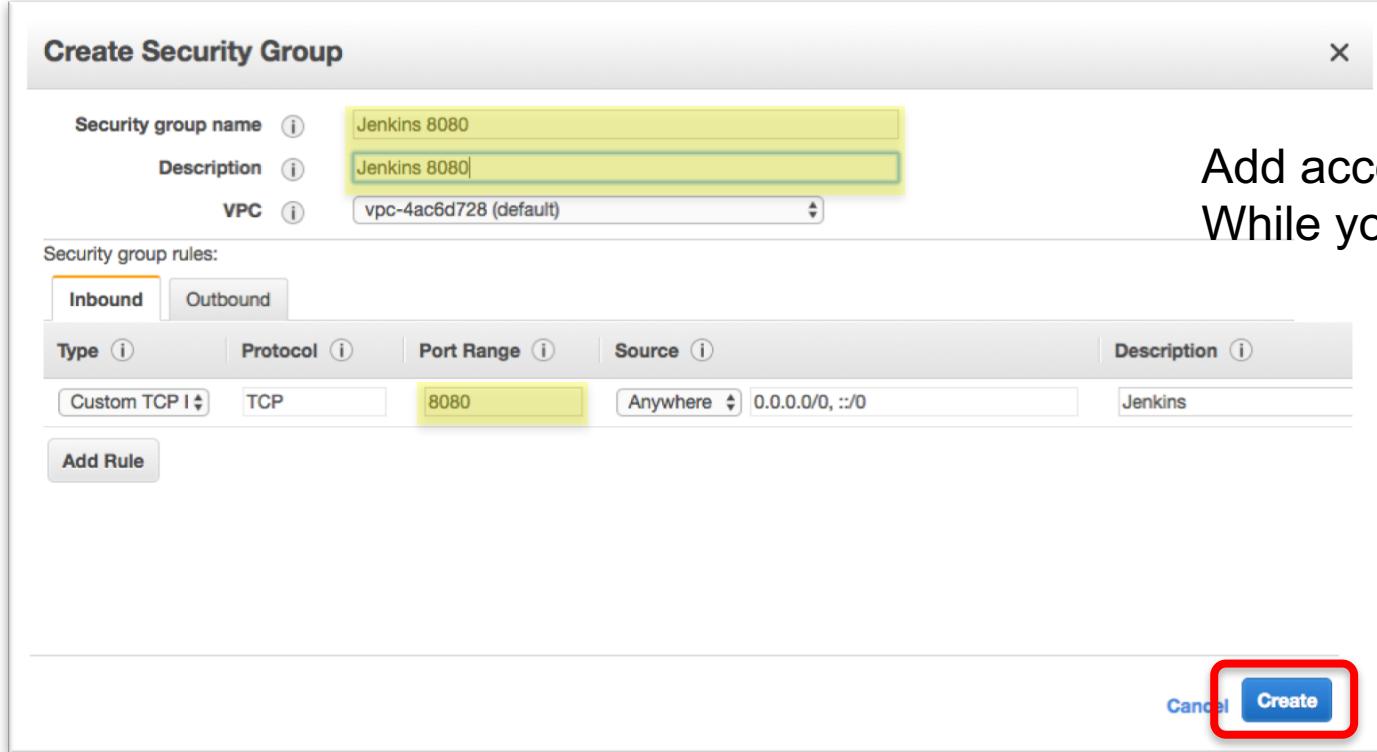
Security group rules:

Inbound    Outbound

Type	Protocol	Port Range	Source	Description
Custom TCP	TCP	8080	Anywhere	0.0.0.0/0, ::/0 Jenkins

Add Rule

Cancel    Create



Add access to port 22  
While you are creating this

# EC2: launch EC2 instance

The screenshot shows the AWS EC2 Dashboard. On the left, there's a sidebar with navigation links: EC2 Dashboard, Events, Tags, Reports, Limits, INSTANCES (with sub-links: Instances, Spot Requests, Reserved Instances, Scheduled Instances, Dedicated Hosts), IMAGES (with sub-links: AMIs, Bundle Tasks), and ELASTIC BLOCK STORE (with sub-links: Volumes, Snapshots). The main content area is titled "Resources" and displays the following information: 0 Running Instances, 0 Dedicated Hosts, 0 Volumes, 0 Key Pairs, 0 Placement Groups, 0 Elastic IPs, 0 Snapshots, 0 Load Balancers, and 3 Security Groups. Below this, there's a promotional message about Amazon Lightsail. At the bottom, there's a "Create Instance" section with the text: "To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance." A large blue button labeled "Launch Instance" is centered in this section, and it is highlighted with a red rectangular box.

# AWS: launch EC2 instance

The screenshot shows the AWS EC2 Dashboard. On the left, there's a sidebar with navigation links like EC2 Dashboard, Events, Tags, Reports, Limits, Instances, Spot Requests, Reserved Instances, Scheduled Instances, Dedicated Hosts, AMIs, Bundle Tasks, and Elastic Block Store. The main area is titled 'Resources' and shows usage statistics for the US West (Oregon) region: 0 Running Instances, 0 Dedicated Hosts, 0 Volumes, 0 Key Pairs, 0 Placement Groups, 0 Elastic IPs, 0 Snapshots, 0 Load Balancers, and 3 Security Groups. A callout box suggests trying Amazon Lightsail. Below this is a 'Create Instance' section with a 'Launch Instance' button highlighted by a red box.

The screenshot shows the 'Step 1: Choose an Amazon Machine Image (AMI)' screen. It's part of a six-step wizard. The first step is 'Choose AMI'. The 'Quick Start' section includes 'My AMIs', 'AWS Marketplace', and 'Community AMIs'. A checkbox for 'Free tier only' is available. The main list displays three AMI options: 1. 'Amazon Linux AMI 2017.09.0 (HVM), SSD Volume Type - ami-e689729e' (selected, highlighted with a red box), 2. 'Red Hat Enterprise Linux 7.4 (HVM), SSD Volume Type - ami-9fa343e7', and 3. 'SUSE Linux Enterprise Server 12 SP3 (HVM), SSD Volume Type - ami-8a887f12'. Each item shows its provider (Amazon Linux, Red Hat, SUSE Linux), image type (HVM, SSD), volume type (EBS or EBS General Purpose), root device type (ebs), and virtualization type (hvm). A 'Select' button is present for each row, with the first one also having a '64-bit' link below it.

# Select a free tier eligible instance type, launch

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
<input type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate
<input checked="" type="checkbox"/>	General purpose	t2.micro	1	1	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.small	1	2	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.large	2	8	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.xlarge	4	16	EBS only	-	Low to Moderate

Cancel

Previous

Review and Launch

Next: C

Step 6: Configure Security Group

Security group is a set of network rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group:

Create a new security group  
 Select an existing security group

Security Group	Name	Description
sg-fd7aed80 awseb-e-gpsjv2ymgz-stack-AWSEBLoadBalancer	Group-1LYQMCTI3FX3UElastic Beanstalk created security group used when no ELB security group is specified.	Elastic Beanstalk created security group used when no ELB security group is specified.
sg-f070e78d awseb-e-gpsjv2ymgz-stack-AWSEBSecurityGroup-This is a temporary security group.	SecurityGroup for ElasticBeanstalk environment.	SecurityGroup for ElasticBeanstalk environment.
sg-e46c7586default	default VPC security group	default VPC security group
sg-ce74e3b3.Jenkins8080	Jenkins8080	Jenkins8080

Inbound rules for sg-ce74e3b3 (Selected security groups: sg-ce74e3b3)

Type	Protocol	Port Range	Source	Description
Custom TCP Rule	TCP	8080	0.0.0.0/0	jenkins port 8080
Custom TCP Rule	TCP	8080	::/0	jenkins port 8080

Cancel Previous Review and Launch

# Create key pair & launch

Select an existing key pair or create a new key pair X

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

**Key pair name**

You have to download the **private key file** (\*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Cancel
Launch Instances

IPv4 Public IP 34.215.204.5

Instance ID	Instance Type	Availability Zone	Instance State	Checks	Alarm Status	Public DNS (IPv4)
66822a015e6f24	t2.micro	us-west-2b	<span>running</span>	Validizing	None	ec2-34-215-204-5.us-west-2.compute.amazonaws.com
560f268cbcfbd2	t2.micro	us-west-2b	<span>terminated</span>		None	ec2-34-215-204-6.us-west-2.compute.amazonaws.com
e6f24 Public DNS: ec2-34-215-204-5.us-west-2.compute.amazonaws.com						 
Details	Monitoring	Tags				
Instance ID	i-04a66822a015e6f24		Public DNS (IPv4)	ec2-34-215-204-5.us-west-2.compute.amazonaws.com		
InstanceState	running		IPv4 Public IP	34.215.204.5		
InstanceType	t2.micro		IPv6 IPs	-		
Private IPs			Private DNS	ip-172-31-33-155.us-west-2.compute.internal		
Zone	us-west-2b		Private IPs	172.31.33.155		
Groups	launch-wizard-2, view inbound rules		Secondary private IPs			
Events	No scheduled events		VPC ID	vpc-4ac6d728		
AMI ID	amzn-ami-hvm-2017.09.020170930-x86_64-gp2 (ami-e689729e)		Subnet ID	subnet-ba0315d8		
Platform	-		Network interfaces	eth0		
IAM role	-		Source/dest. check	True		

# Create new admin Linux user

- Add\_user seconadmin
- Use Vault to generate and store key
- Log out of ec2-user
- Log in as seconadmin  
`ssh -I <vaultgeneratedkey> seconadmin@<ip>`

# ~/.ssh/config shortcut

```
host isaca
  User ec2-user
  Hostname 34.215.204.5
  IdentityFile /Users/trice/Downloads/ISACA_workshop.pem
```

```
$ sudo yum -y update
Loaded plugins: priorities, update-motd, upgrade-helper
amzn-
main
| 2.1 kB 00:00:00
amzn-
updates
| 2.5 kB 00:00:00
Resolving Dependencies
--> Running transaction check
---> Package amazon-ssm-agent.x86_64 0:2.1.4.0-1.amzn1 will be updated
```

# Login!

```
$ chmod 600 ~/Downloads/ISACA_workshop.pem
$ ssh -i ~/Downloads/ISACA_workshop.pem ec2-user@34.215.204.5
Warning: No xauth data; using fake authentication data for X11
forwarding.
X11 forwarding request failed on channel 0
```



```
https://aws.amazon.com/amazon-linux-ami/2017.09-release-notes/
No packages needed for security; 5 packages available
Run "sudo yum update" to apply all updates.
```

# Agenda

- 8:30-9:30 CI/CD overview / AWS and Git prerequisites
- 9:30-10:00 AWS Overview / Fork the GitHub Hello Word Repo
- 10:00:10:30 Provision EC2, Elastic Beanstalk and security groups
- 10:30-10:45 Break
- 10:45 - noon install Jenkins with Git and EB plugins
- break
- Noon-1:30 lunch
- 1:30 – 2:00 Vault overview
- 2-3:00 vault install and config
- 3-4 instrument Jenkins with vault secrets
- 4-5 build and deploy Hello World
- 5-5:30 wrap up and best practices



**HashiCorp**  
**Vault**

A Tool for Managing Secrets

# Why Vault?

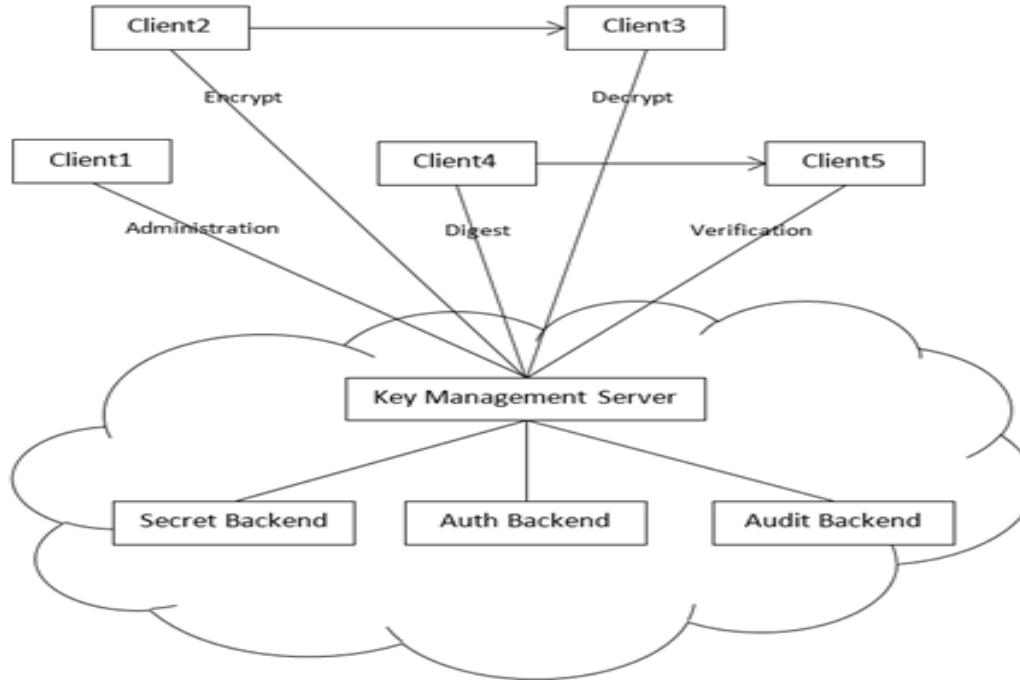
- To leverage a single secure storage to store secrets such as API keys, passwords, certificates, and more for cloud applications.
- Unified interface, while ensuring strict controls and recording audit logs.
- Hardware HSMs though provides stronger controls are very expensive!!
- Ensures secrets are always stored separately from the data

\*secret = passwords, tokens, private keys, API keys, symmetric encryption keys

# Weak links in Cloud



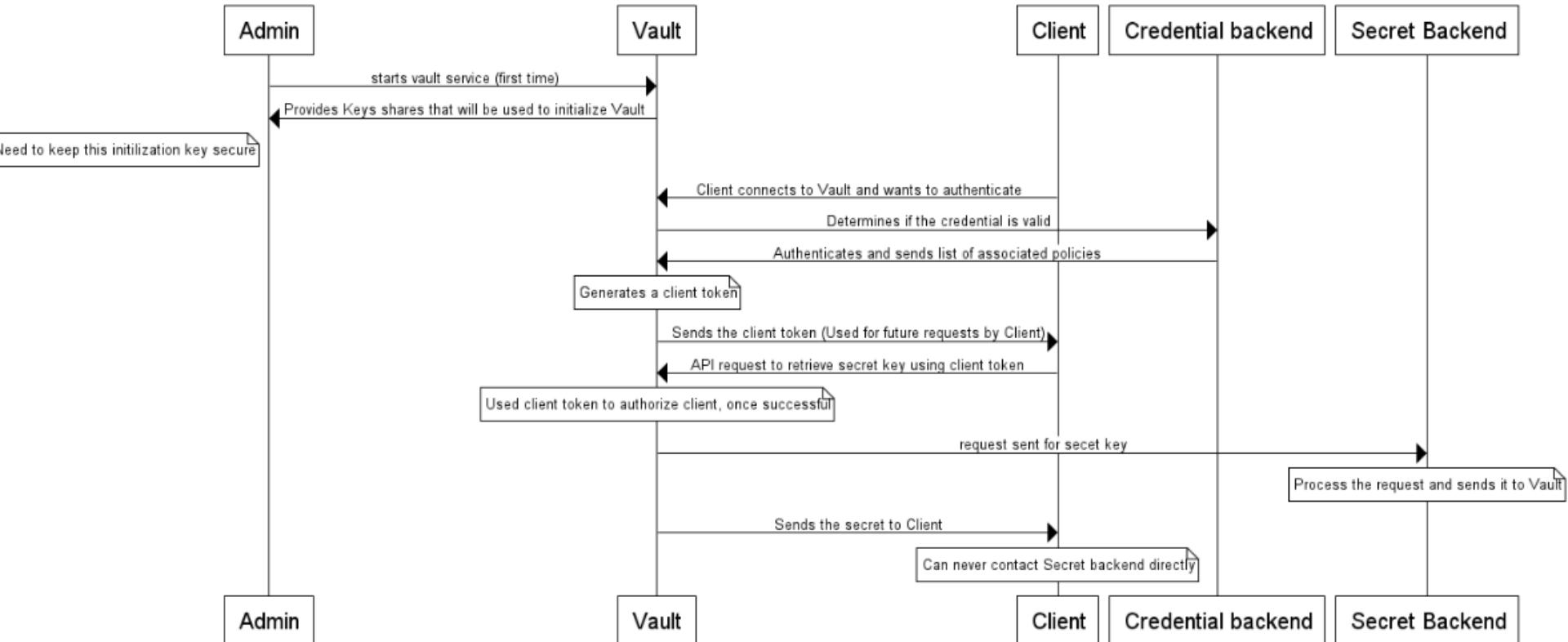
# Key Management System overview



# Key features of KMS

- Key generation
- Key Storage
- Key retrieval
- Policies for key rotation
- Manipulation of keys (activation date or description of key, revocation, delete)
- Access control or management
- Redundancy - mechanism for real time mirroring, load balancing, high availability
- Audit access - audit trail

# Vault high level example



# Vault key strategies

- Deploy up-to-date vault version (v0.6.1 or above)
- Securely initialize and bring up the KMS
  - Protection of master keys that are needed to initialize the KMS
- Support High Availability
- Secure Communications Between Clients and KMS
- Authentication using automated ways (Tokens, PKI Keys)
- Roles and privileges should be well defined so that clients are granted least amount of privileges necessary to complete its task.

- More info on Vault can be found at the Infosec recipe –  
<https://cisco.jiveon.com/docs/DOC-1510893>

Centralized Key and Secret Management Like

Document created by **Khadija Amin** on Jun 30, 2016 • Last modified by **Khadija Amin** on Feb 20, 2017

	<b>Chef:</b>	khadija Amin (khamin), Yasi Xi, Sanchuan Cheng
	<i>Sous Chef(s):</i>	John Qian

# Setup vault

```
19-5 ~$ ./vault init
161160 [INFO ] core: security barrier not initialized
165330 [INFO ] core: security barrier initialized: shares=5 threshold=3
504942 [INFO ] core: post-unseal setup starting
523053 [INFO ] core: loaded wrapping token key
523210 [INFO ] core: successfully setup plugin catalog: plugin-directory=
529808 [INFO ] core: successfully mounted backend: type=kv path=secret/
529985 [INFO ] core: successfully mounted backend: type=cubbyhole path=cubbyhole/
530172 [INFO ] core: successfully mounted backend: type=system path=sys/
544246 [INFO ] expiration: restoring leases
544499 [INFO ] rollback: starting rollback manager
545759 [INFO ] expiration: lease restore complete
550890 [INFO ] core: post-unseal setup complete
551190 [INFO ] core/startClusterListener: starting listener: listener_address=127.0.0.1:8
551353 [INFO ] core/startClusterListener: serving cluster requests: cluster_listen_addres
558804 [INFO ] core: root token generated
558815 [INFO ] core: pre-seal teardown starting
558820 [INFO ] core: stopping cluster listeners
558825 [INFO ] core: shutting down forwarding rpc listeners
558840 [INFO ] core: forwarding rpc listeners stopped
551861 [INFO ] core: rpc listeners successfully shut down
551895 [INFO ] core: cluster listeners successfully shut down
551912 [INFO ] rollback: stopping rollback manager
551978 [INFO ] core: pre-seal teardown complete
:vb8jHeC9H9p2N4uEVcAYDspbT3z0Z4HuamXQS
:CXKjpbnSfHeq3Xub2/Bc1lYEXkGfC5tTSkabI
:ivjs6MD+ZG0K1pjrwTo/PHJ7BbpRggSjxEqd
:8nJeZ11vAQMMLdx-fx56elwDbdiAr6Zwo4Ler
:7/dhE38LionuHScdkZcCD+NJAfffZ7YxeM5r
:9e9ca13-5157-1543-6469-e24a9604be3a

:th 5 keys and a key threshold of 3. Please
:the above keys. When the vault is re-sealed,
:, you must provide at least 3 of these keys
```

the master key. Without at least 3 keys, it is permanently sealed.

```
Vault is now sealed.
[ec2-user@ip-172-31-19-5 ~]$ ./vault status
Sealed: true
Key Shares: 3
Key Threshold: 3
Unseal Progress: 0
Unseal Nonce:
Version: 0.8.3

High-Availability Enabled: true
    Mode: sealed
[ec2-user@ip-172-31-19-5 ~]$ ./vault unseal
Key (will be hidden):
Sealed: true
Key Shares: 5
Key Threshold: 3
Unseal Progress: 1
Unseal Nonce: ae5f8f64-4162-d360-d267-9c05c6828f0a
[ec2-user@ip-172-31-19-5 ~]$ ./vault unseal
Key (will be hidden):
Sealed: true
Key Shares: 5
Key Threshold: 3
Unseal Progress: 2
Unseal Nonce: ae5f8f64-4162-d360-d267-9c05c6828f0a
[ec2-user@ip-172-31-19-5 ~]$ clear
[
[ec2-user@ip-172-31-19-5 ~]$ ./vault unseal
Key (will be hidden):
2017/10/31 21:06:45.106334 [INFO] core: vault is unsealed
2017/10/31 21:06:45.106401 [INFO] core: entering standby mode
2017/10/31 21:06:45 [INFO] agent: Synced check 'vault:127.0.0.1:8200:vault-sealed-check'
Sealed: false
Key Shares: 5
Key Threshold: 3
Unseal Progress: 0
Unseal Nonce:
[ec2-user@ip-172-31-19-5 ~]$ 2017/10/31 21:06:45.115051 [INFO] core: acquired lock, enabling
2017/10/31 21:06:45.148488 [INFO] core: post-unseal setup starting
2017/10/31 21:06:45.149037 [INFO] core: loaded wrapping token key
2017/10/31 21:06:45.149046 [INFO] core: successfully setup plugin catalog: plugin-directory=
2017/10/31 21:06:45.149649 [INFO] core: successfully mounted backend: type=kv path=secret/
2017/10/31 21:06:45.149743 [INFO] core: successfully mounted backend: type=system path=sys/
2017/10/31 21:06:45.149757 [INFO] core: successfully mounted backend: type=cubbyhole path=cub
```

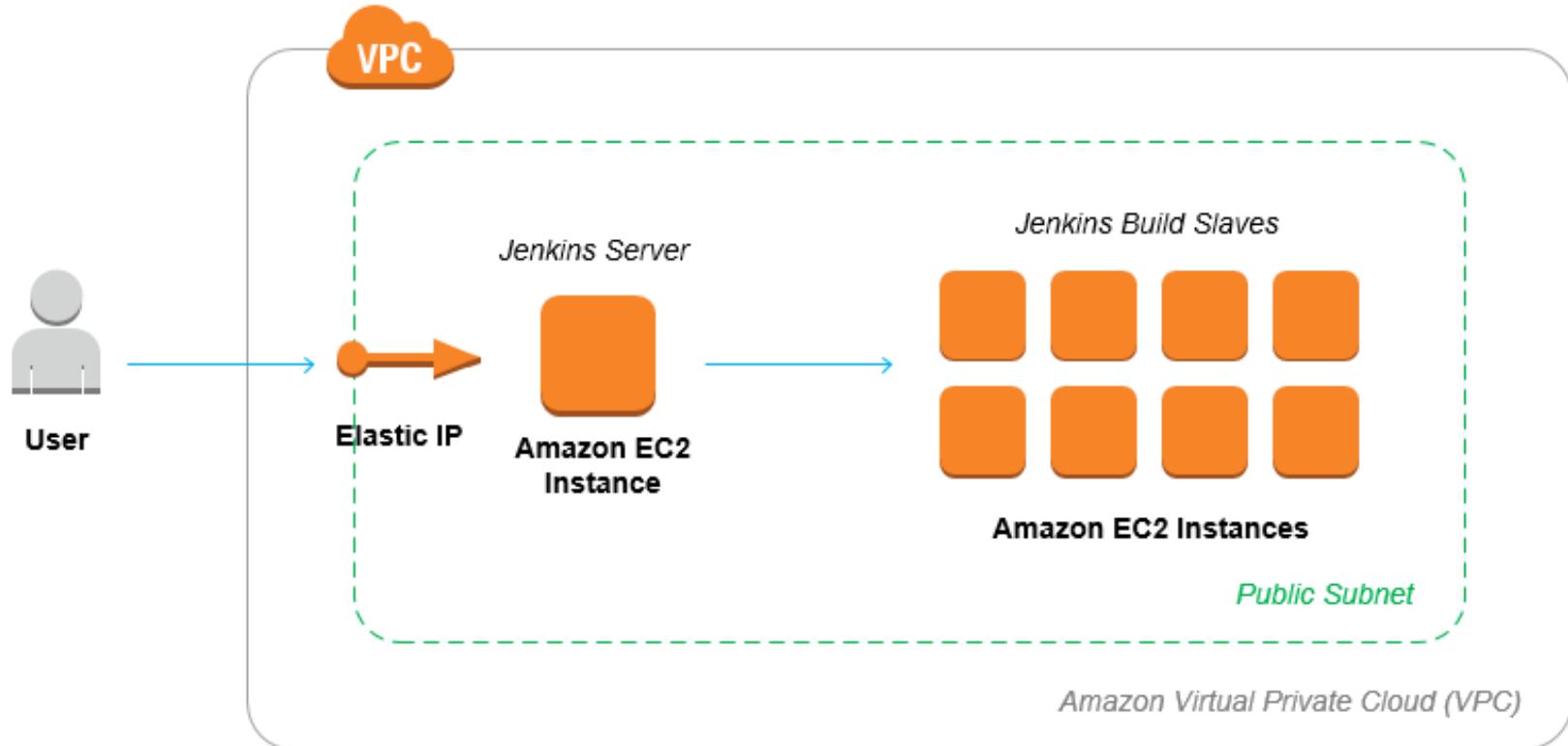
# Add a secret

From your Jenkins/Vault server

```
curl http://127.0.0.1:8200/v1/auth/github/login  
-d '{ "token": "YourGitHubPersonalToken" }'
```

```
curl -H "X-Vault-Token: client_token" -X POST -d '{  
"gittoken": "YourPersonalToken" }'
```

# Jenkins



# Jenkins: installation

```
$ sudo yum update -y  
$ sudo yum install -y git
```

```
$ sudo wget -O /etc/yum.repos.d/jenkins.repo \  
http://pkg.jenkins-ci.org/redhat/jenkins.repo  
$ sudo rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key  
$ sudo yum install -y jenkins
```

```
$ sudo service jenkins start  
Starting Jenkins
```

[ OK ]

# If you get Java8 errors

```
$ sudo service jenkins start
Starting Jenkins Jenkins requires Java8 or later, but you are running 1.7.0_151-
mockbuild_2017_08_09_21_42-b00 from /usr/lib/jvm/java-1.7.0-openjdk-1.7.0.151.x86_64/jre
java.lang.UnsupportedClassVersionError: 51.0
at Main.main(Main.java:124)
$ sudo yum -Y install java-1.8.0
$ sudo yum -Y remove java-1.7.0-openjdk
$ sudo service jenkins start
Starting Jenkins
$
```

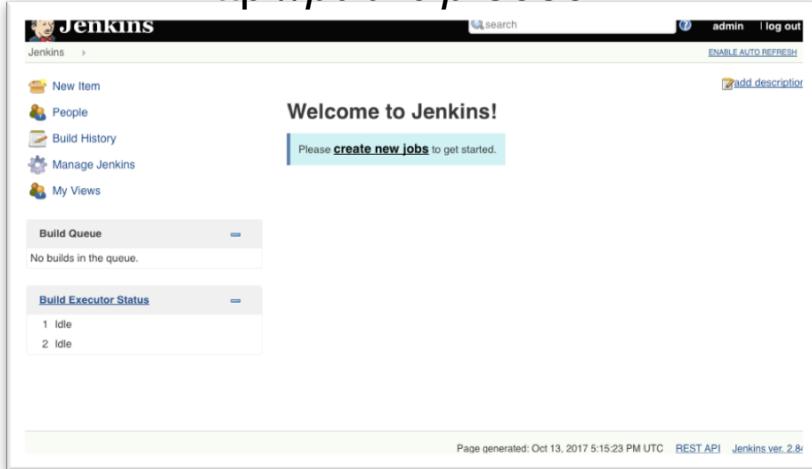
[ OK ]

# Jenkins: Login

From EC2 Linux prompt

```
$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword  
somestring
```

*http://publicip:8080*



# Create a build job – Embed vault secret

```
#!/bin/bash --x

VAULTTOKEN=(curl -s http://127.0.0.1:8200/v1/auth/github/login -d '{ "token": "YOURVAULTTOKEN" }' |
python -c 'import sys, json; print json.load(sys.stdin)[ "auth" ][ "client_token" ]')

GITTOKEN=(curl -s -H "X-Vault-Token: $VAULTTOKEN" -X GET http://127.0.0.1:8200/v1/secret/seccon |
python -c 'import sys, json; print json.load(sys.stdin)[ "data" ][ "gittoken" ]')

if [ -d "hello-world" ];
then
cd hello-world
git pull
else
git clone https://$GITTOKEN:x-oauth-basic@github.com/rtpphokie/hello-world.git
fi
rm -Rf hello-world
```

*Before you type... there's a copy in the github repo*

# Create a build job – Vault Plugin

## Vault Plugin

Vault URL

Vault Credential

Vault Secret

Path /secret/seccon

Environment Variable

Key name

With Ant

Jenkins Credentials Provider: Jenkins

Domain

Kind

Scope

Personal Access Token

ID

Description

*Before you type... there's a copy in the gitlab repo*

# Create a build job – Use secret to fetch code

```
#!/bin/bash -x

if [ -d "hello-world" ]; then
    cd hello-world;
    git pull
else
    url = https://$MYSECRET:x-oauth-basic@github.com/rtpphokie/hello-world.git
    git clone $URL
fi
rm -Rf hello-world
```

*Before you type... there's a copy in the github repo*

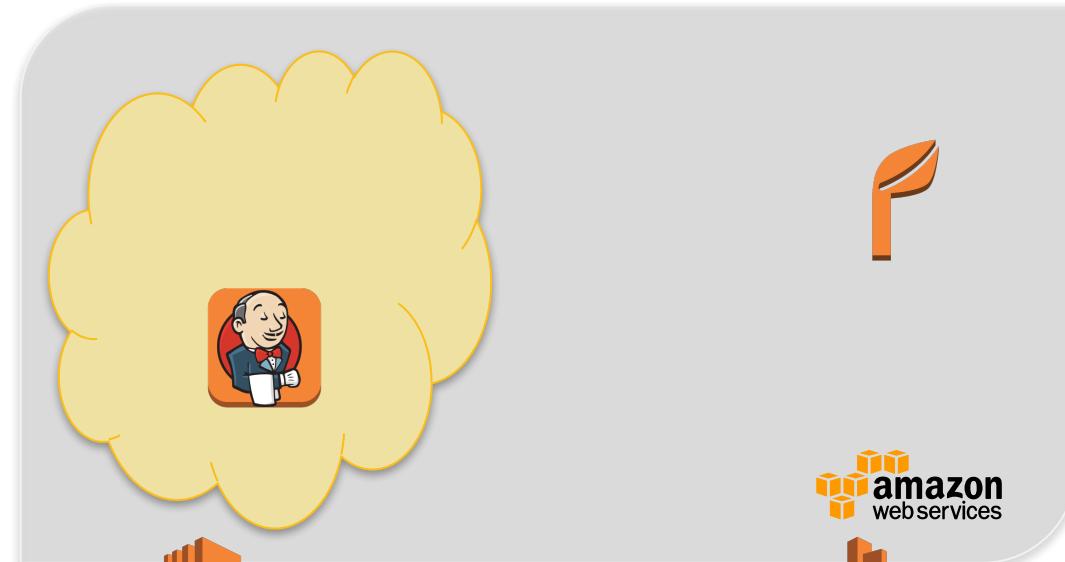
# Our Continuous Delivery Pipeline

Source Code  
Management



GitHub

Build



Identity



Deployment



Elastic  
Beanstalk

# Create a key pair (Github – Jenkins)

- Login to EC2 instance
- Switch to the Jenkins user  
`sudo su - jenkins`
- Generate a key  
`ssh-keygen -t rsa`
- Public and private `is_rsa` files are in `~jenkins/.ssh`

# From the Jenkins GUI: Create a build job

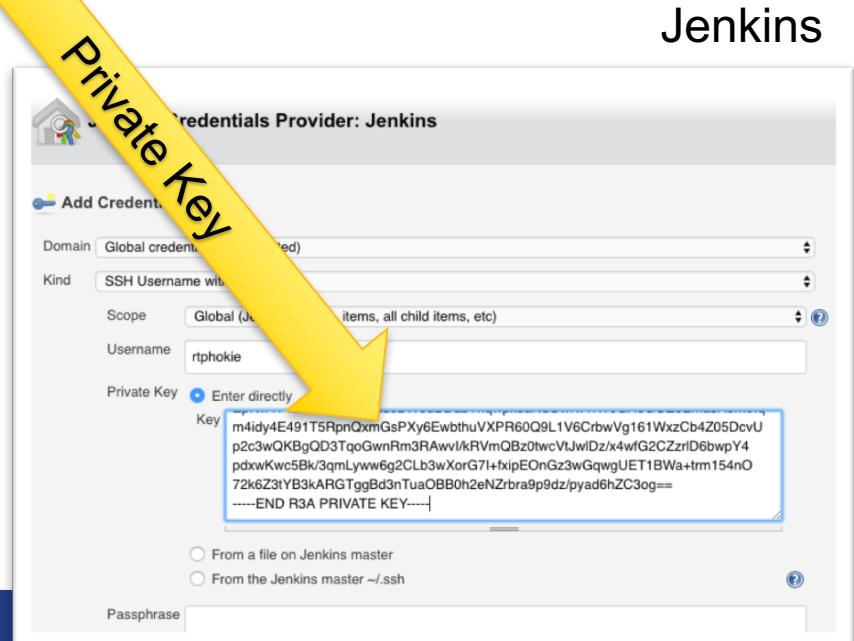
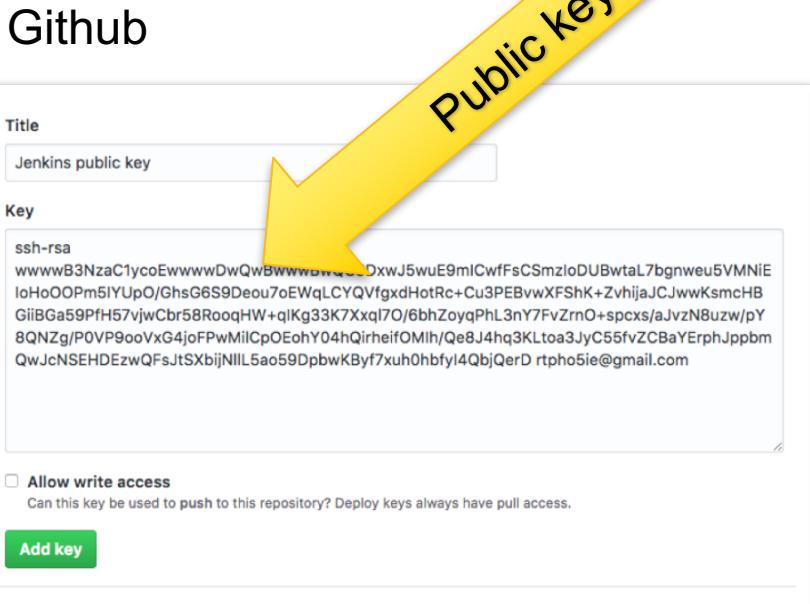
The screenshot shows the Jenkins interface for creating a new item. In the top left, there's a search bar with the placeholder "Enter an item name" and a text input field containing "HelloNCAA". Below it, a note says "Required field". To the right, a large text box contains the instruction "Click New Item, select Freestyle project".

The main area shows the configuration for a "Freestyle project". It includes:

- Source Code Management:** A section where "Git" is selected as the provider. It shows fields for "Repository URL" (http://rtp) and "Credentials".
- Build:** A section for AWS Elastic Beanstalk settings:
  - AWS Credentials and Region: Fields for "Credentials" (AKIAJBOP35EQ67KDT2AA), "AWS Region" (us-west-2), and "Number Of Attempts" (30). A "Validate Credentials" button is present.
  - Application and Environment: Fields for "Application Name" (My First Elastic Beanstalk Application) and "Environment Name(s)" (Default-Environment). A "Validate Coordinates" button is present.

# Github: Deploy key

```
$ ssh-keygen -t rsa -C 'trice@cisco.com'
```



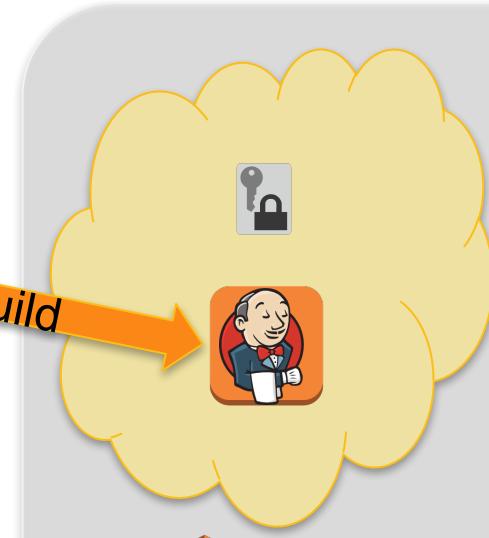
# Our Continuous Delivery Pipeline

Source Code  
Management



GitHub

Build



Identity



Deployment



Elastic  
Beanstalk

# Jenkins: deploy to EB post build

The image shows two overlapping windows. The background window is the Jenkins AWS Elastic Beanstalk configuration screen, and the foreground window is the Jenkins Credentials Provider interface.

**Jenkins AWS Elastic Beanstalk Configuration (Background):**

- AWS Credentials and Region:**
  - Credentials: AKIAJYHV6HXF567KPADA (selected)
  - AWS Region: us-west-2
  - Number Of Attempts: 30
- Application and Environment:**
  - Application Name: helloncaa
  - Environment Name(s): helloncaa-env
- Version and Deployment:**
  - Version Label Format: \${BUILD\_TAG}

**Jenkins Credentials Provider: Jenkins (Foreground):**

Select AWS Credentials

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain: Global credentials (unrestricted)

Kind: AWS Credentials

Scope: Global (Jenkins, nodes, items, all child items, etc)

ID: (empty)

Description: (empty)

Access Key ID: (empty)

Secret Access Key: (empty)

IAM Role Support: (empty)

Yellow arrows point from the 'AWS Region' field, the 'Application Name' field, and the 'Environment Name(s)' field to their respective fields in the Jenkins Credentials Provider interface.

# Artifact Management

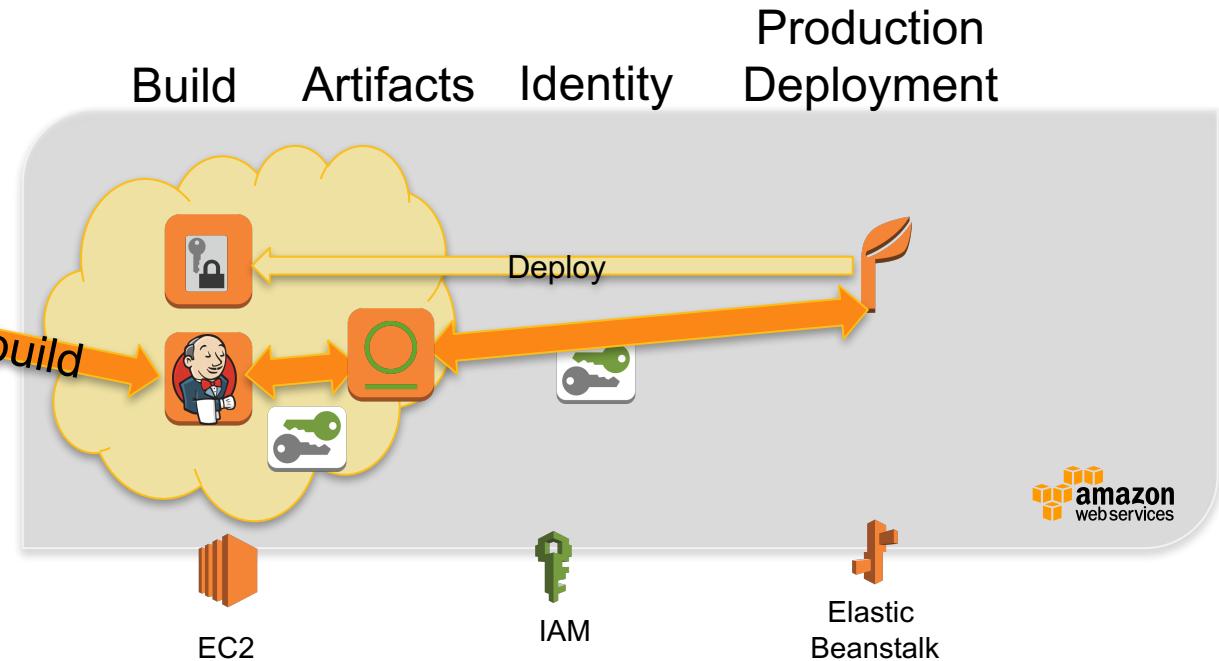
Source Code  
Management



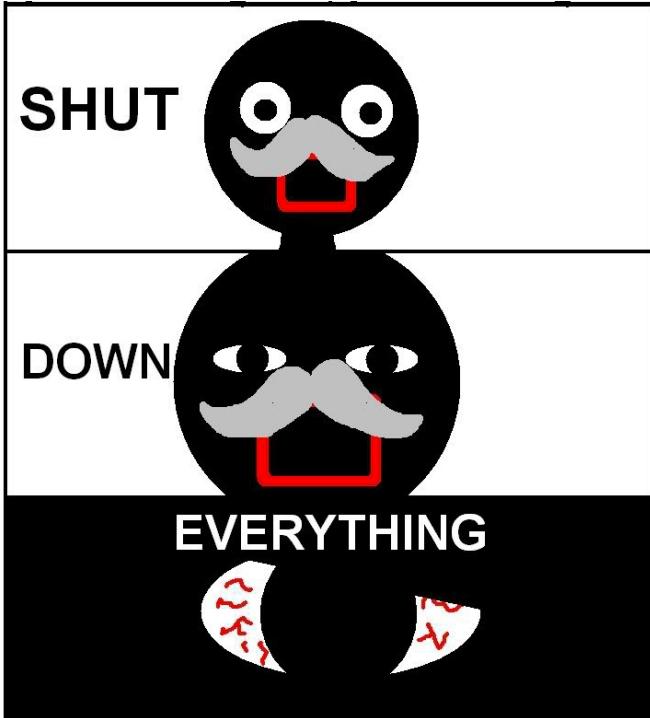
Trigger build



GitHub



# Before you go



Avoid unexpected bills:

**Terminate all EC2 instances and Elastic Beanstalk apps.**

Thank you.

[trice@cisco.com](mailto:trice@cisco.com)



 @rtphokie

