AWS Meetup: DevOps

Date: 5/26/2016

Presenter: Aater Suleman

Flux7

About Flux7

Aater Suleman
Co-Founder & CEO Flux7
Faculty, University of Texas at Austin

Flux7: Cloud and DevOps Solutions
Founded in 2013
Team of 35+
Headquartered in Austin, Texas

Achievements

AWS DevOps, Healthcare, and Life Sciences Competencies

TechTarget's "**Impact Best AWS Consulting Partner**" two years in a row (2015 & 2016)

Partner Recognition Award by AWS at reInvent 2015

Customers featured on stage at AWS re:Invent three years in a row

Docker Foundation and authorized consulting partner

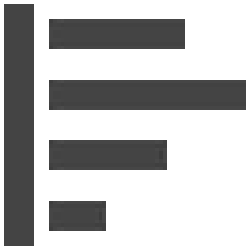
150+ happy customers through word of mouth

"[Flux7] taught us how to do 10x the work in 1/10th the time" - Patrick K, AWS Re:invent'14, CTO's Keynote

Flux7

Quick Poll

HOW MANY?



- ★ Frontend HTML/JS developers
- ★ Backend developers
- ★ Operations folks
- ★ Business: Managers/executives

HOW MANY?

- ★ Enterprise (> 1B in cap)
- ★ Mid-tier
- ★ SMBs

POP

Purpose:

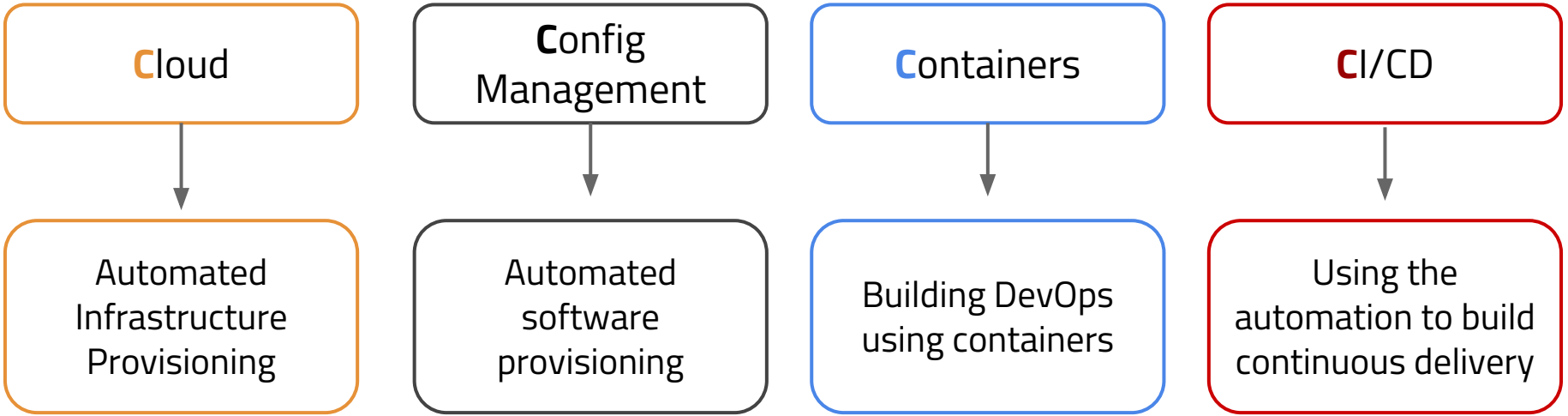
To provide the audience working knowledge and a sample DevOps workflow implemented on AWS

Outcomes:

- The audience shall be able to:
  - The role of 4Cs in DevOps
  - A sample use of each C and how everything ties together
- Take home a working DevOps environment

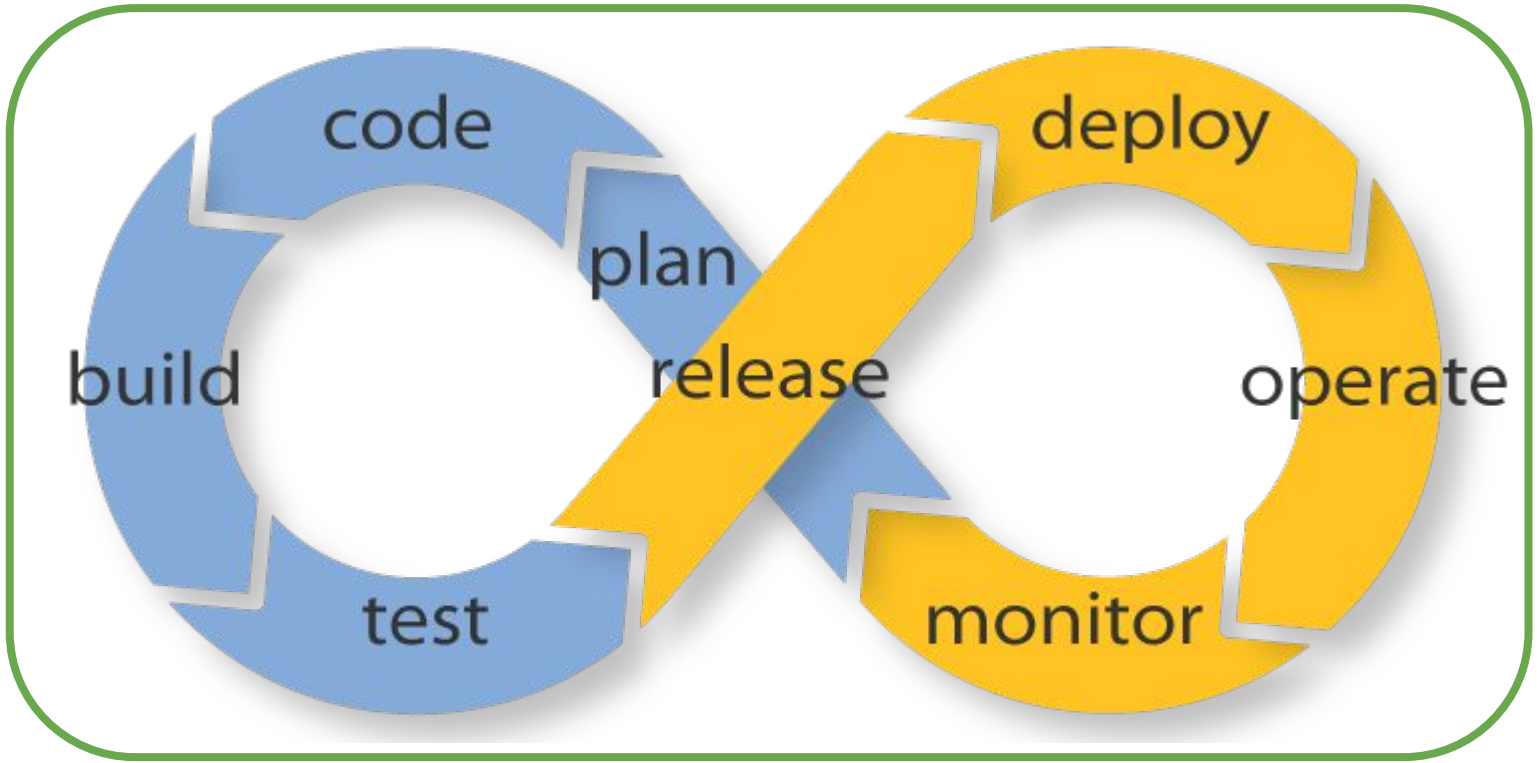
POP (Continued)

Plan:



DevOps

DevOps



Delivery of Technology

Delivering technology includes the delivery of:

Infrastructure

Code

Server Configurations

Delivery of Technology

Delivering technology includes the delivery of:

Infrastructure

Code

Server Configurations

AWS DevOps 4 Cs - Part 2

Configuration Management

Pre-reqs

A laptop with web browser, a text editor, and Wifi

AWS account with PowerUser privileges

Basic understanding of CloudFormation for provisioning EC2 instances

Outcomes

You will be able to:

- ☑ Understand where CloudFormation::Init fits in the picture
- ☑ Read and modify CloudFormation::Init code
- ☑ Deploy and Debug an application stack with CF::Init
- ☑ Test stack with a Hello World application

Agenda

Present

- ➔ Provide a big picture view
- ➔ Concepts
- ➔ Walkthrough sample code

Hands On

- ➔ Deploy a Stack
- ➔ Extend the deployment

What is Configuration Management?

Name	Examples
App	**App, Assets, App config**
Config	**Prerequisite software for app (IIS, Apache, Monitoring agents, etc)**
OS	**AWS-provided**

Last meetup: [Slides and notes in Github]

CF::Init

A mechanism to provision software on an EC2 instance

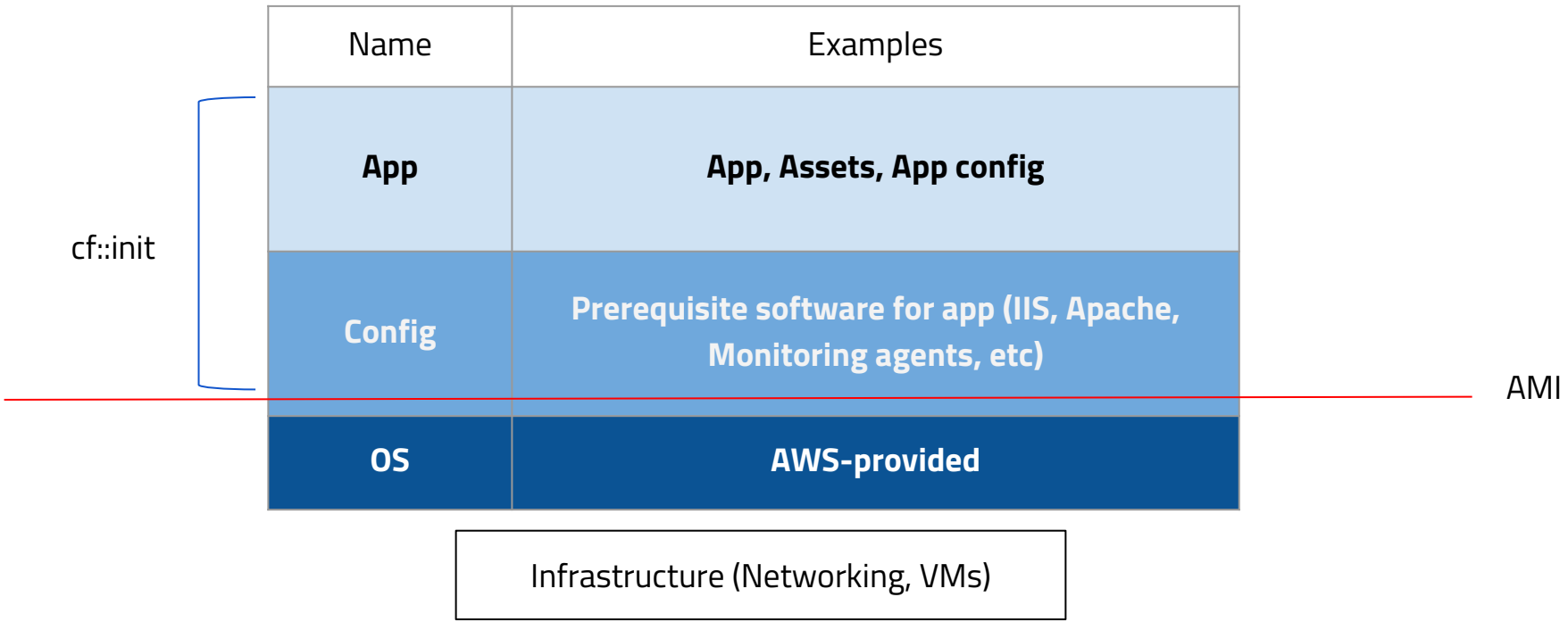
Typically used in conjunction with CloudFormation at the time of instance creation

Provides hooks for typical configuration tasks: install packages, download archive files, and create users, groups, files, folders.

Capable of running arbitrary shell commands or scripts

Leads to immutable infrastructure and configuration fully specified in a single file

Where does it fit in?



CF::Init vs. Chef/Puppet

CF::Init

Requires an agent

Uses resources in the CloudFormation stack itself as configuration DB

A part of AWS CloudFormation

Allows for provisioning and maintenance

Chef/Puppet

Requires an agent

Uses a centralized server/service as configuration DB

3rd party independent tool

Allows for provisioning and maintenance

CF::Init vs. Chef/Puppet

CF::Init

Designed for “cattle”

but can be used with “pets” with effort

Focus on provisioning

Tight integration with infrastructure provisioning

Auto-scaling and spot-instance friendly

Chef/Puppet

Designed for “pets”

but can be used with “cattle” with effort

Focus on management

Richer and more programmable

Larger ecosystems

Can be used outside of AWS

Other solutions from AWS

OpsWorks:

Chef-based solution for configuration management

ElasticBeanstalk:

Pre-built AMIs (aka. VM images) for common platforms like node.js, Django, IIS, etc

CF::Init

CloudFormation::Init

- ✓ Mechanism for configuring an EC2 instance at launch (or later)
- ✓ Describe the configuration with a JSON descriptor

```
"AWS::CloudFormation::Init" : {
    "configSets" : {
        "set1" : [ "1" ]
    },
    "1" : {
        "commands" : {
            "test" : {
                "command" : "echo \"$CFNTEST\" > test.txt",
                "env" : { "CFNTEST" : "I come from config1." },
                "cwd" : "~"
            }
        }
    }
}
```

Terms

Metadata:

Specified structured data with a CF resource

User data:

A mechanism to specify a “boot” script in EC2

cfn-init:

Agent to install software and to start services

More Terms

Config:

A collection of steps specifying what to do on an instance

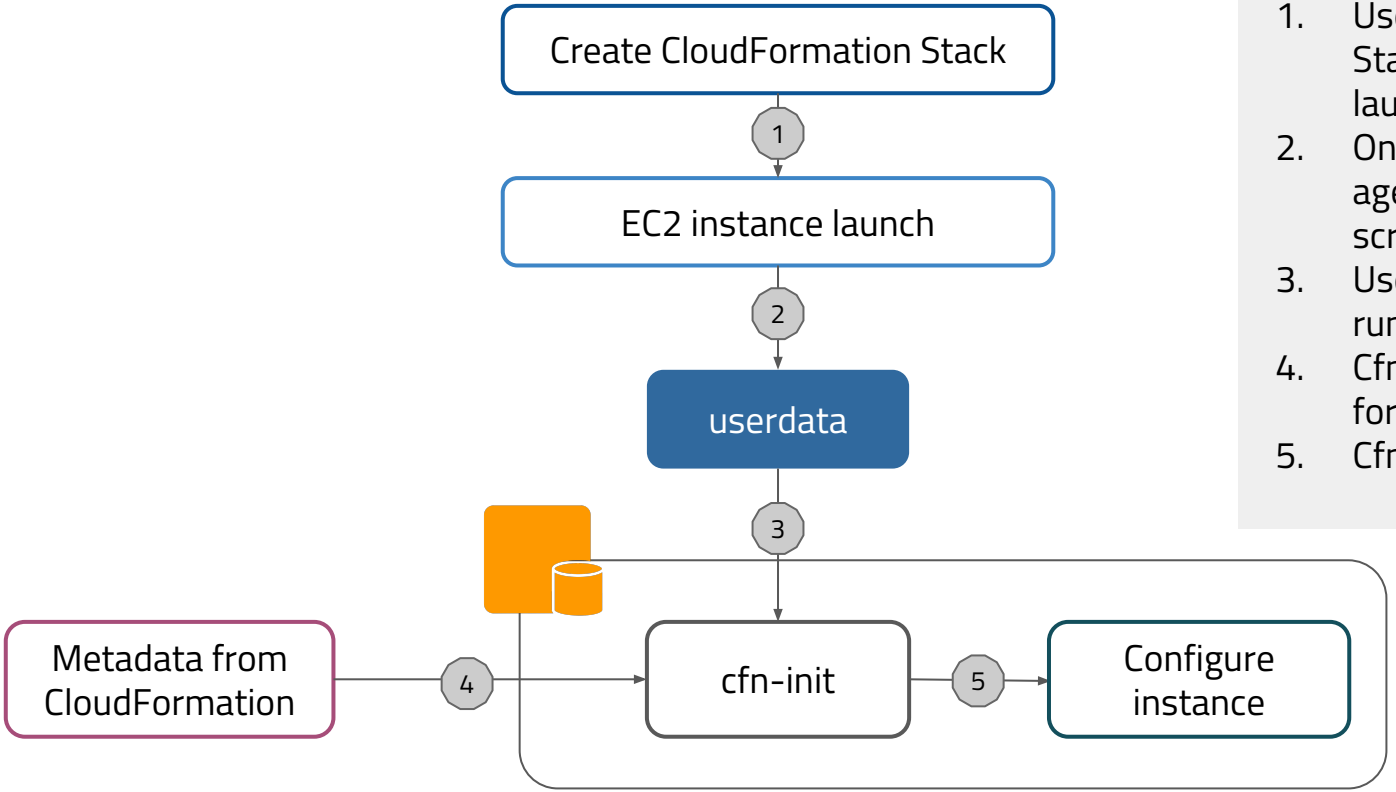
configSet:

A set of configs to run on the instance

Resource:

The EC2 instance being configured

How it works?



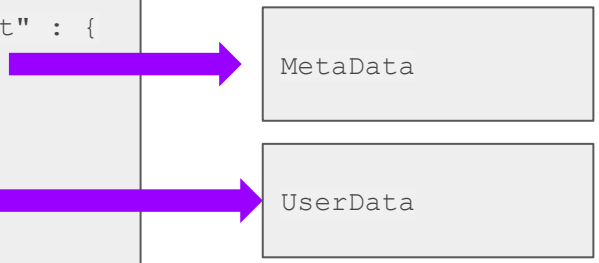
1. User triggers creation of a CF Stack which triggers an EC2 launch
2. Once launched, cloud-init agent on the instance runs a script specified in user data
3. User data script installs and runs cfn-init
4. Cfn-init downloads metadata for resource from CF
5. Cfn-init runs config sets

Flux7

How it is specified

```
"MyInstance": {
    "Type": "AWS::EC2::Instance",
    "Metadata" : {
      "AWS::CloudFormation::Init" : {

      }
    },
    "Properties": {
    }
  }
```

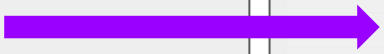


Flux7

UserData

```
"MyInstance": {
    "Type": "AWS::EC2::Instance",
    "Metadata" : {
    },
    "Properties": {
      :

    }
  }
```



```
      "UserData"       : { "Fn::Base64" : { "Fn::Join" : ["", [
              "#!/bin/bash -xe\n",
              "yum update -y aws-cfn-bootstrap\n",

              "# Install the files and packages from the metadata\n",
              "/opt/aws/bin/cfn-init -v ",
              "         --stack ", { "Ref" : "AWS::StackName" },
              "         --resource WebServerInstance ",
              "         --configsets Install ",
              "         --region ", { "Ref" : "AWS::Region" }, "\n"
            ]]}}
    }
```

Resource: The name of the EC2 resource which has the metadata

configsets: Which configsets specified in the meta data to run (more to come)

Flux7

MetaData

```
"MyInstance": {
    "Type": "AWS::EC2::Instance",
    "Metadata" : {
      "AWS::CloudFormation::Init" : {

      }
    },
    "Properties": {
    }
  }
```

```
"AWS::CloudFormation::Init" : {
    "configSets" : {
        "ascending" : [ "config1" , "config2" ],
        "descending" : [ "config2" , "config1" ]
    },
    "config1" : {
      Config Attributes
    },
    "config2" : {
      Config Attributes
    }
}
```

Flux7

Configs

commands:

Commands to run

packages:

Packages to install

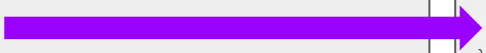
users:

Users to create

& a few more

How it is specified?

```
"AWS::CloudFormation::Init" : {
    "configSets" : {
        "ascending" : [ "config1" , "config2" ],
        "descending" : [ "config2" , "config1" ]
    },
    "config1" : {
      Config Attributes
    },
    "config2" : {
      Config Attributes
    }
}
```



```
"commands" : {
    "test" : {
        "command" : "echo \"$MAGIC\" > test.txt",
        "env" : { "MAGIC" : "An env" },
        "cwd" : "~",
        "test" : "test ! -e ~/test.txt",
        "ignoreErrors" : "false"
    }
},
"packages" : {
    "yum" : {
        "httpd" : [],
        "php" : [],
        "wordpress" : []
     }
},
"users" : {
    "myUser" : {
        "groups" : ["groupOne", "groupTwo"],
        "uid" : "50",
        "homeDir" : "/tmp"
    }
}
```

Anything else?

- **Sources**: Download and unpack an archive like tar, zip, tar.gz, etc
- **Groups**: Create Unix/Linux groups and assign Ids
- **Files**: To create a new file with inline content or from a URL
- **Services**: To start services on Linux or Windows

Resources

Starter Code:

https://github.com/Flux7Labs/aws-devops-tutorial

http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-init.html

http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/updating.stacks.walkthrough.html

Thank You

How do I:

? **Write a template**
- ◎ Text editor (VS, Eclipse)

? **Catch syntactic errors**
- ◎ CF validate

? **Catch logical errors**
- ◎ ChangeSets

? **Provision**
- ◎ Create stack

? **Access outputs**
- ◎ Review the console

? **Update an existing stack**
- ◎ Update stack

? **Debug errors**
- ◎ Review error logs

CF::Init vs. CodeDeploy

CF::Init

Runs when VM is created

Can run arbitrary code

Best for installing immutable components often universal across applications and change with “infrastructure,” e.g., Anti-Virus, Monitoring agent, etc.

CodeDeploy

Runs after VM is running, including scale up events

Can run arbitrary code

Best for installing components that are application specific and change between application deploys, e.g., Application code, config files, Apache modules, etc