12B

DevOps Background

• Dev + Operations organizations

• Disconnect between dev and operations leads to inefficiencies

• Dev wants change vs. Ops want stability

• Communication, collaboration and integration

• DevOps is about people, process and technology

• Agility in all areas of SDLC

• Agility during development

• Agility during integration

• Agility during testing

• Agility deploying applications

• Agility monitoring and correcting issues in infrastructure

• Fast and responsive ops in sync with pace of dev teams

• Automated integration & testing

DevOps Toolset

• Puppet, Chef, Ansible are some of common tools

• Automated & repeatable infra configuration

• CI / CD tools tools like Jenkins, Cruise Control

• Automated Monitoring like Nagios, Icinga2

• Automated tools for testing SonorQube

DevOps Culture Shift

• Cross-disciplinary approach

• “We are all in it together” feeling – no us vs. them

• Better system reliability

• Combined (Dev, QA and Ops) Scrum sessions

• Ops think – how to build and test infrastructure

• Dev think – Efficient way of deploying – Managing prod systems

12C AWS CLI

* What is AWS CLI?
  + • **Unified command line toolset** to manage AWS resources
  + • Automate start, stop and management of resources
  + • Alternate to AWS Console
  + • Most AWS services are supported
  + • Windows, Linux and Mac OS X support
* Need both CLI and SDK (Software Development Kit)
  + • AWS Console is most common way to start services
  + • Can be tedious to start hundreds of instances
  + • CLI can help automate these – Wrap in shell scripts for efficiency
  + • AWS SDK can provide additional flexibility managing multiple services
    - – Help with dependencies
    - – Wait for one service to complete before kicking off another
    - – Programmatically using queues (SQS), Event driven programming (AWS Lambda)
* Installing AWS CLI
  + • Written in Python (version 2.7+)
  + • MSI installer is an option on Windows
  + • Python Pip recommended
  + • Bundled installer (zip file)
* • Configuring AWS CLI
  + %> aws configure AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE AWS Secret Access Key [None]: wJalrXUtnFEMI/ K7MDENG/bPxRfiCYEXAMPLEKEY
  + Default region name [None]: us-west-2
  + Default output format [None]: ENTER
  + %> aws configure –-profile jdoe AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE AWS Secret Access Key [None]: wJalrXUtnFEMI/ K7MDENG/bPxRfiCYEXAMPLEKEY
  + Default region name [None]: us-west-2
  + Default output format [None]: ENTER
* • Working with AWS CLI for EC2
  + List all EC2 instances – %> aws ec2 describe-instances
  + • Create EC2 instance and print InstanceID – %> aws ec2 run-instances --image-id ami-29ebb519 --count 1 --instance-type t2.micro --key-name devenv-key --securitygroups devenv-sg --query 'In stances[0].InstanceId’
  + • Stop EC2 instance – %> aws ec2 stop-instances --instances-id i-1a2b3c4d 10 AWS Automation using CLI
  + • List all S3 buckets – %> aws s3 ls
  + • Create S3 bucket – %> aws s3 mb s3://my-bucket
  + • Copy file to S3 bucket – %> aws s3 cp local-file s3://mybucket
  + • Sync local directory to S3 bucket – %> aws s3 sync local-dir s3://my11 bucket
* • Working with AWS CLI for S3

12E

Cloud Automation using AWS SDK

• AWS SDK (Software Development Kit) Intro

• Allows developers to write software that interacts with AWS services

• Support for IDE tools such as Eclipse

• Languages supported

– Java – Python – Ruby – .NET – JavaScript / Node.js – PHP – Mobile SDK

• Credentials for AWS SDK

• Using AWS SDK for Python

AWS Python SDK

• Based on Boto3

• Allows us to create and manage AWS resources using Python

• Services supported – EC2 – S3 – AutoScaling – ELB – CloudFront

Sample tasks with Boto3

• Start / Stop instances

• Register / De-register instances from ELB

• Create snapshots

• Allocate and assign EIP

• Create and manage SQS

• Create and assign EBS volumes

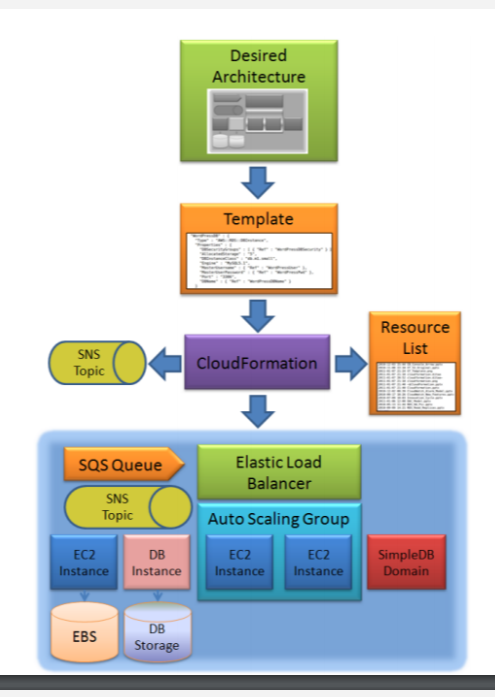
• Take snapshots of EBS

12F Cloud Automation with Ansible

• Introduction

* Ansible Introduction
  + • Configuration Management
  + • Provisioning tool
  + • Deployment tool
  + • Open Source and commercial version
  + • Agent-less
  + • Idempotent
  + • Uses SSH to manage instances
  + • No complicated server installation
  + • Simpler than Chef, Puppet
* Ansible Inventory file
  + • Ansible executes playbooks on a group of nodes
  + • Nodes are specified in a file called inventory
  + • One host per line
  + • Host can be grouped together: webservers, app, db etc
  + • Inventory is ini style
* Ansible Modules
  + • Modules perform a specific action
    - • install package, run command, create file etc
    - • Also called tasks because they do the actual work
  + • Over 200+ modules included with Ansible
  + • Modules are equivalent to chef resources
  + • Custom Modules can be written in any language
* Ansible Playbook
  + • File that includes group of tasks or modules
  + • Playbooks are written YAML (Yet Another Markup Language)
  + • Multiple playbooks are normally applied to a host
  + • Each playbook is designed to achieve one task
    - • Install apache server
* Ansible Template
  + • Use templates to create dynamic files
  + • Application config files, database name, user name / password etc
  + • Variables will be substituted at run time
  + • Jinja2 format
  + • Variables, conditionals, loops etc
* Ansible Vault
  + • Allows you to store sensitive data
  + • Encrypts passwords, license keys using AES-256

AWS CLOUD FORMATION WITH AWS CLOUDFORMATION

* • Need for automating infrastructure build
* • Overview of AWS CloudFormation (CF)
  + How to provision complex infra?
  + • Some options
    - – Write build scripts
    - – Provisioning tools like BladeLogic
    - – Combination of scripts + manual setup
  + • Challenges
    - – Developing and maintaining repeatable code
    - – Catch provisioning errors
    - – Wait for individual components to come up or order of provisioning
    - – Override behavior for different environments
    - – How to handle changes to infrastructure (additional components, upgrades etc)
* • Benefits of CF
  + • Create JSON-based template
  + • CF will create all the AWS resources you requested including
    - – Security groups – EC2 – S3 – ELB – Route53 – RDS – …
    - • Templates can be modular, shared and re-used
    - • Stacked templates
  + • Orderly and predictable way of creating AWS resources
  + • Support for conditionals – If US-East-1 region, use this ami
  + • Wait for components to come up completely
  + • Pass parameters to template or use defaults
  + • Console and CLI version support
  + • Rollback from failures
  + • Resubmitting updated templates causes only delta (changes) to be applied
  + • Deleting stack deletes all resources created by template
  + BENEFITS:
    - • Define AWS resources
    - • Manage relationships between AWS resources
    - • Lookup resources that were elsewhere in the template using Ref tag
    - • Conditionals help you to design data-driven infrastructure
    - • Parallel stack processing
    - • Diff between existing resources and updated template – Apply minimum changes required
  + Ex.
* • CF Templates
  + • Template version and Description
  + • Resources – AWS Resources
  + • Parameters – pass runtime variable values, default values (optional)
  + • Mappings – Conditionals
  + • If this region => then that AMI
  + • If this region => then this SSH key
  + • Outputs – Return values to caller (optional)
* • Template structure
* • Sample Templates
* • Updating / Deleting stacks – Template modifications
  + • Applications rarely stay static once they are in production
  + • Typical tasks are – changing security group – adding new instances
  + • Template can be resubmitted to CF service
  + • Note: its best practice to version control your templates
* • CloudFormer
  + • What if you have existing application?
  + • What if you need to duplicate app in another region?
  + • What if you don’t want to write templates from scratch?
  + • CloudFormer app / service can be used to pick the existing AWS resources
  + • CloudFormer spits out a CF template
  + • Can be used as a starting point with minimal modifications
* • CF alternatives
  + • visualops.io
  + • Terraform.io
* • CF Integration
  + • CF can be integrated with configuration management tools like Chef or Ansible
  + • Specify list of playbooks or cookbooks to run once instance is up (run list)
  + • Install apps from S3, GitHub etc