DevOps Essentials DevOps and the Cloud

DevOps and the Cloud

* DevOps and the Cloud are not the same thing
* DevOps – a culture of collaboration between Dev and Ops
* The Cloud – remote servers on the internet that offer services in place of locally-hosted solutions. “The cloud is someone else’s computer”
* The culture and practices of DevOps are very useful in the world of the cloud
* DevOps and the Cloud developed alongside one another, and many cloud services are built on DevOps practices.
* The can also be a great tool for DevOps. Many cloud services offer features that support DevOps practices.

Traditional Non-Cloud Stack

* A traditional stack is a regular, self-hosted datacenter
* In a traditional stack, you are responsible for every layer of the architecture, and you provide all of the infrastructure necessary to run your apps.
  + Application
  + Data
  + Runtime
  + Middleware
  + O/S
  + Virtualization
  + Servers
  + Storage
  + Networking
  + You are responsible for cost, maintaining, making sure they stay up. Everything because it is yours

Infrastructure as a Services

* With IaaS, someone else provides the low-level infrastructure
* The cloud service provider gives you a bare OS
* You are responsible for all installation and configuration above the OS level
* Examples
  + EC2
  + Microsoft Azure VMs and containers
  + Google Compute Engine

Platform as a Service

* With PaaS, everything below the application and Data layers is abstracted.
* The cloud service provider gives you a way to deploy and app and use databases
* You are only responsible for managing the app and data
* Examples
  + AWS Elastic Beanstalk
  + Heroku
  + Google App Engine

Software as a Service

* With SaaS, everything is managed.
* The cloud service provider gives you an application ready for use
* You are only responsible for using the application
* Examples
  + G-mail
  + Microsoft Office 365

Serverless

* Serverless is also known as Function as a services (FaaS)
* Serverless is different from the traditional application architecture
* Everything is abstracted. You deploy small, single-purpose functions.
* You pay for the compute resources used by your functions
* Examples
  + Lambda
  + Azure functions
  + Google Cloud Functions

DevOps and Google Cloud Platform

GCP DevOps features

* Google App Engine: Paas – deploy your code, and don’t worry about the rest. Comes with built in support for microservices and out-of-the-box autoscaling. Certain configurations can be considered serverless.
* Google Compute Engine: IaaS – Deploy and orchestrate clusters of VMs on Googles architectures. Build-in orchestration and works very closely with App Engine. Can be managed with other tools like Ansible, Salt, Puppet, and Chef.
* Google Cloud Functions: FaaS – Quickly and easily create and deploy FaaS functions.
* Google Cloud SDK: software development kit used for interacting with GCP APIs. Makes it easy to build your own tools and automations that interact with GCP.
* Stackdriver: Monitoring – monitoring, logging, and diagnostics for your GCP services. It also works with AWS.
* Cloud Deployment Manager: Declarative configuration for your GCP stack. Allows for IaC and automated deployment. YAML based.
* Google Kubernetes Engine: Orchestration on GCP with Kubernetes. You can also do continuous integration with Jenkins on Kubernetes engine.

DevOps and Microsoft Azure

* Visual studio Team Services – source control and CI
* Jenkins – CI for Java apps
* Continuous Deployment Triggers – automated deployment triggers integrated with CI.
* Orchestration: Azure Container Registry – repository of container images
* Azure Container Service – Kubernetes orchestration
* Azure Web Apps – Cloud hosting for web apps integrated with DevOps pipeline
* Monitoring: Azure Application Insights – APM, diagnostics, and analytics. Supports machine learning.
* FaaS/Serverless: Azure Functions – autoscaling, serverless functions in Azure

DevOps and AWS

* Amazon EC2: IaaS, easily scalable, and offers you full control over your cloud infrastructure.
* Integrates with tons of tools, both AWS and 3rd party
* AWS Elastic Beanstalk: PaaS, offers out-of-the-box load balancing and autoscaling, but you can still access underlying AWS resources with full control.
* CI/CD and deployment:
* AWS CodeBuild – CI
* AWS CodeDeploye – CD
* AWS CodePipeline – full code pipeline from build to deploy
* AWS CodeStar – integrates all parts of the process with project management tools and JIRA issue tracking
* IaC: CloudFormation – stack templating engine configured in YAML or JSON
* OpsWorks – IaC with chef
* Serverless/FaaS – AWS Lambda – run serverless functions on AWS
* Monitoring – Amazon CloudWatch – track metrics and logs, set alarms, and automate responses to monitoring data