Certified DevOps Professional – Notes

**Introduction:**

* $300 and has a 30-day retake policy
* Can only take it 3 times in 12 months
* 170 Minutes long (2.1 minute per question)
* 80 questions
* Questions are long
* Appropriate level of information in the key exam areas
* Will only cover up to 70% of exam. Practical expertise and general knowledge. DevOps Knowledge
* Re-watch all videos multiple times. Read, YouTube videos, white papers, practice!

**Core Concepts:**

* AWS CLI/API
* SLDC (Software Development Lifecycle)
* Continuous Integration, Build, delivery and deployment
  + Continuous Integration (CI) & Continuous Deployment (CD)
  + Problems with GIT occurred by changes made by other developers being incompatible and causing compile failures. Known as integration hell.
  + Longer the code was checked, the greater issue could arise with more compile issues
  + **Continuous Integration** – process of automating regular code commits followed by an automated build and test process designed to highlight integration issues early.
    - Requires additional tools like Bamboo, Cruise Control, Jenkins, Go and Team City
    - Customizable Workflow based integration
    - Spot prices could be a good fit for compilation and testing servers.
  + **Continuous Deployment** – takes the form of a workflow based process which accepts a tested software build payload from a CI server.
    - CD Server automates the deployment into a working WA, Pre-Production or production environment.
    - CodeDeploy and CodePipeline provide CI/CD services. Same with Elastic Beanstalk and Cloudformation.
  + Developers -> Source -> Build -> Staging -> Production -> Customers – Ideas Requests Bugs -> Changes Update Fixes
* Deployment Types
  + **Single Target Deployment**
    - Not used great these days
    - Mainly for Small development projects, legacy or non-highly available HA infrastructure
    - Build -> Target
    - Brief outage when version installed. Testing limited. Rollback involves removing the new version and installing the previous one.
  + **All-at-once deployment**
    - Happens in one step but destination is multiple targets
    - More complicated than single target, requiring orchestration tooling.
    - No ability to test, more for small deployments. Small outages and less than ideal rollback.
  + **Minimum in-service style deployment**
    - Happens in multiple stages
    - Deployment occurs to as many targets as possible while maintain minimum targets
    - Moving parts with orchestration and health checks required
    - No downtime
    - Quicker and less stages
    - Allows automated testing, targets assessed and tested prior
  + **Rolling deployment**
    - Happens in multiple stages but user defines number of targets
    - Moving parts with orchestration and health checks required
    - Applicable health isn’t necessarily maintained.
    - Can be least efficient deployment based on time-taken
    - Allows automated testing, targets assessed and tested prior
    - No downtime
    - Can be paused allowing limited multi-version testing.
  + **Blue/Green deployment**
    - Requires advanced orchestration tooling
    - Extra cost
    - Rapid deployment process.
    - Cutover/migration can be clean (DNS Change)
    - Rollback (DNS regression)
    - Health and performance of entire green field can be tested
    - Can be fully automated via template systems.
    - Binary
  + **Know Pro/Cons for Exam**
  + **Know when each should be used and when not**
  + **Know the limitations of each, how quick deployment, how quick rollback**
  + **Know how each deployment type impacts your applications**
  + **Known which AWS service support deployment type.**
* A/B Testing
  + **Sends a percentage of traffic to green/blue environment**
  + Separates different versions of your code.
  + Can allow testing/feedback to come from users
  + Allows gradual performance/stability/health analysis
  + New features can be tested.
  + Uses Route53 with 2 records in simple mode. Later switched to weighted.
    - DNS, caching, other DNS related issues can impact overall accuracy of technique.
* Bootstrapping
  + **Bootstrapping** – process during which you start with a base image, ISO/AMI, and automation build on it to create a more complex object.
  + CFINIT or CLOUDINIT
  + AMI based approach would require a lot of AMI’s.
  + Bootstrap can be done via cloud formation.
  + Help to bring all the components together like a cake.
  + Quick launch versions AMI’s use pre-built AMI’s with minimal configuration changes.
* Immutable Architecture
  + Immutable Architecture – practice of replacing infrastructure instead of upgrading or repairing faulty components.
  + Treat servers as unchangeable objects
  + If something develops a problem, diagnose, fix and return to service.
  + Treats servers as throwaway objects. If a failure happens, remove the server and create a new one from an AMI.
  + Never work manually.
  + Traditional architecture is like pets. Immutable servers are like cattle.
* Containers & Docker Primer
  + **Virtualization**
    - Guest OS, Dependencies, Application, VM
    - Wasted space
  + **Containerization**
    - Dependencies, Appliance, Container.
    - Higher density and improved portability
    - Escape from dependencies.
    - Consistent progression from Dev->Test->QA-Prod
    - **Isolation** – performance or stability issues with App A in Container A, won’t impact App B in Container B
    - Resource scheduling at the micro level.
    - Code portability
    - Micro-Services
  + **Docker Components**
    - **Docker Image** – basis of a docker container ISO. Read only.Base Build docker containers.
    - **Docker Container** – holds everything needed to make an application to work.
    - **Layers/Union File System** – Combines layer into a single image. Branches are separate file systems.
    - **Docker File** – instructions create or include each layer. Stored in a docker file.
    - **Docker Daemon/Engine** – create OS to run your applications. Communicates with the docker client to build/ship/run containers
    - **Docker Client** – interface between you and the engine. Control docker daemon
    - **Docker Registries /Docker Hub** – hold images in a repo. Provided by Docker Hub. Can use images based on what others have done.
* JSON Primer
  + **JSON (JavaScript Object Notation)** – way to represent structured data for interchange between appliances.
  + Used most often with Web services like Rest API
  + **Name/Value pairs** – consists of key followed by a value
  + Can be a string, array, object, null value, JSON structure
  + **Object** – collection of key/value pairs.
  + **Array** – ordered list of values surrounded by values.
  + **JSON String** – contains an array of values or an object
  + **Policy document (JSON string)** – complicated information contained by nested objects.

**CI/CD/Automation:**

CloudFormation Primer

* **CloudFormation** – building block service designed to provision infrastructure (cfn)
  + ElasticBeanstalk using cloudformation
  + JSON based. **Know how to read/write JSON.**
  + Let cfn name the resources used in the template.
  + **Stack** –cloud formations unit of grouping for infrastructure. Controls lifecycle of the infrastructure.
    - Has stack ID. Can be applied many times.
  + **Template** – JSON document giving cloud formation instructions on how to act and what to create. Used to update or create.
    - Template limit of 200.
  + **Stack Policy** – IAM style policy which governs what can be changed and by who. (cfnStack)
  + **Parameters –** allow the passing of veriables into a template via UI, CLI, or API.
  + **Mappings –** allow processing of hash’s (arrays of key value pairs) by the cfnTemplate.
    - Use case – define lookup to select ami id based on region.
  + **Resources –** where your actual resources are declared.
    - Required in the template.
  + **Outputs –** results from the template.
  + Can run scripts within the instances
  + Can have conditional elements to resources or whole resources conditional.
  + Can expand files within instances.
  + Always challenge yourself with infrastructure configuration.