CO328: DevOps
Tutorial 04

E/15/123: Wishma Herath
E/15/280: Pubudu Premathilaka
E/15/316: Suneth Samarasinghe

1. Explain how DevOps extends agile principles to the entire systems lifecycle.

DevOps is primarily an extension of Agile principles to include systems and processes. By comparing Agile and DevOps “DevOps extends agile principles” statement can be figured easily.

- Agile values are expressed with Agile manifesto; The DevOps Values are expresses with Agile Manifesto focusing on service and value delivered to customer/end-user.
- DevOps principle broadly follows agile with the inclusion of systems and operations required to deliver software functionality continuously to users instead of stopping at code check-in.
- DevOps methods extended implementation of agile methods such as Scrum with operations, Kanban with operations, etc. The entire delivery value chain integrated into one agile system.

With DevOps, which extended beyond the "infrastructure as code," configuration management, metrics and monitoring systems, computing tools approach, virtualization and cloud to accelerate the change in the world of modern infrastructure. DevOps brings some tools to the cluster, such as configuration manager (Dummy, Chef, Ansible, Cuffingin), Synchronization (Garden Guard, Noah, Mesos) and virtual monitoring and containers (AWS, Open Stack, Hobo, Port Agent) and many more.

So DevOps is not an independent concept but just a graceful extension to include operations also in the definition of a graceful multi-functional team, working together and working as a team with one goal to deliver the client program completely.

2. What are the similarities and differences between centralized and distributed version control systems?

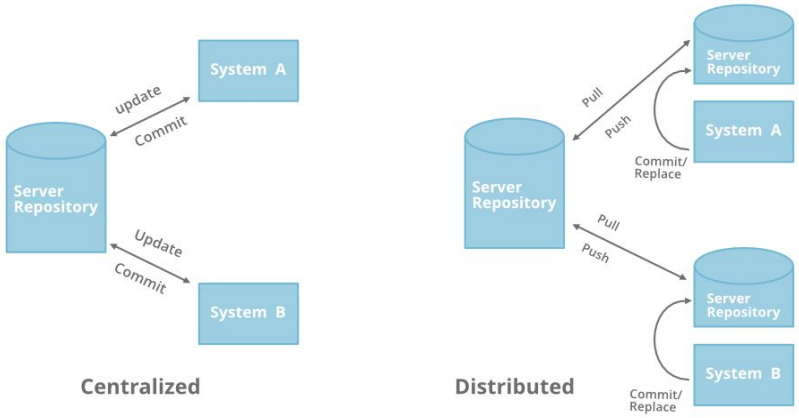
Differences

- The concept of a centralized system is that it works on a Client-Server relationship. The repository is located in one place and provides access to many clients. In a Distributed System, every user has a local copy of the repository in addition to the central repo on the server-side.
- CVCS is dependent on the access to the server whereas DVCS provides the benefits to work offline. Everything except push and pull the code can be done without an internet connection.
- CVCS is easy to understand whereas DVCS has some complex process for beginners.
- DVCS is comparatively fast comparing to CVCS

Similarities

- Both allow users to keep track of the changes in software development projects, and enable them to collaborate on those projects.
- Both help to prevent concurrent work from conflicting.
- if the user accidentally deletes code or a file the user can get it back, or the user can compare previous versions to see why a new bug has crept in. It's also good if one person working in multiple locations.

A clear difference can be found from the following figure



3. Compare at least three configuration management tools. What are their similarities and differences?

1. CFEngine

   CFEngine is a popular open-source configuration management system. Its primary function is to provide automated configuration and maintenance of large-scale computer systems.

   - Created in 1993 by Mark Burgess
   - The first configuration manager
   - Major update in 2009, Cfengine 3
   - Proprietary configuration language
   - Asymmetric Key Encryption
   - Written in C

2. Puppet

   Puppet, an automated administrative engine for your Linux, Unix, and Windows systems, performs administrative tasks (such as adding users, installing packages, and updating server configurations) based on a centralized specification.

   - Created in 2006 by Puppet Labs
   - The easiest solution
   - Proprietary declarative language
   - Modular configuration
   - Asymmetric Key Encryption
   - Ruby-based

3. Chef

A systems integration framework, built to bring the benefits of configuration management to your entire infrastructure.

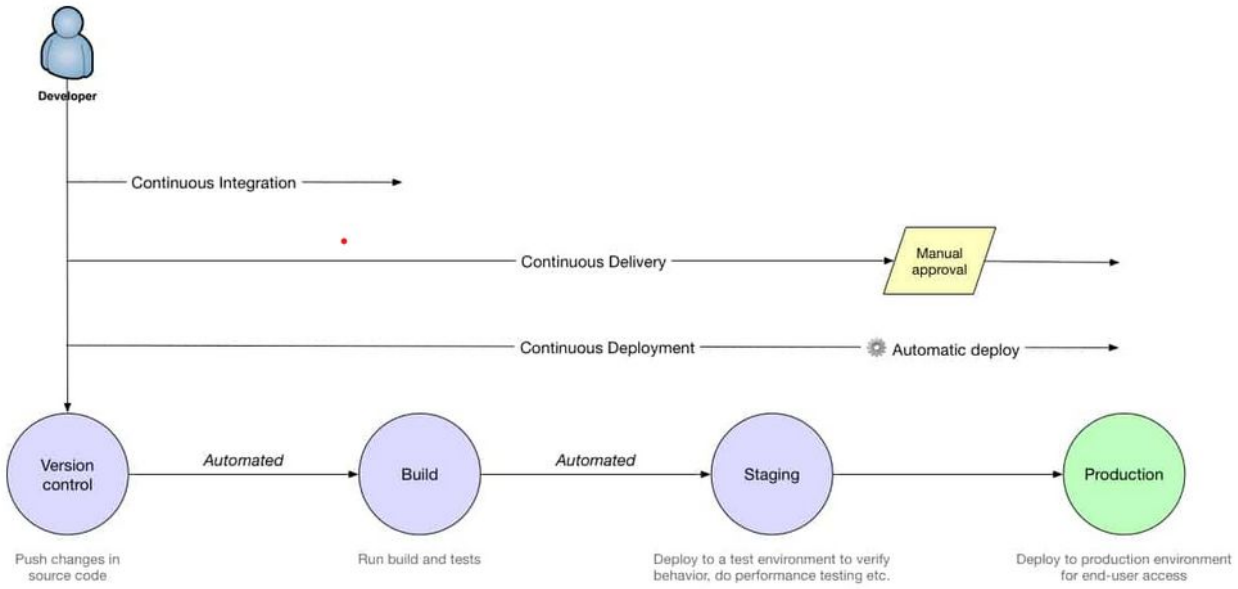
- Created in 2009 by Opscode
- Sustained development
- Configuration language: Ruby
- Modular configuration
- Asymmetric Key Encryption

Summarized comparison of each configuration management tool is displayed below

	Puppet	**Chef**	**Cfengine**
Pull	Yes	Yes	Yes
Push	No	No	No
Idempotence	Yes	Yes	Yes
Config language	Declarative	Ruby	Declarative
Web UI	Yes (limited)	Yes	No
OS Support	Linux/Unix - Windows (experimental	Linux/Unix - Windows (experimental	Linux/Unix - Windows (experimental
License	GPL v2	Apache	GPL
Company	Puppet Labs	OpsCode	Cfengine
Cloud	Yes	SaaS platform	Yes

4. Explain the difference between continuous integration and continuous deployment.

	Continuous Integration	Continuous Deployment.
By definition	Developers merge their changes back to the main branch as often as possible	Extension of continuous integration to release new changes to your customers quickly in a sustainable way
Functionally	Changes are validated by creating a build and running automated tests against the build.	Every change that passes all stages of your production pipeline is released to the customers
Features	The testing/validating process is automated	The release process is automated
	Accelerate testing/validating in the development process	Accelerate the feedback loop with your customers



5. Using at least one example explains the difference between IAAS and PAAS cloud services.

IAAS stands for "infrastructure as a service." It refers to cloud-based infrastructure resources that are delivered to organizations via virtualization technology that helps organizations build and manage their servers, network, operating systems, and data storage.

IaaS Examples

1. Amazon Web Services (AWS)

   AWS is overseen by Amazon and is used for on-demand cloud computing and purchased for on a recurring subscription basis. AWS helps companies store data and deliver content -- in fact, it's helping you read this blog post right now.

2. Microsoft Azure

   Microsoft Azure is a cloud-computing IaaS product that allows for building, testing, and managing applications through a network of Microsoft data centers.

PAAS stands for "platform as a service." It refers to cloud-based platform services that provide developers with a framework they can use to build custom applications upon.

PaaS Examples

1. Google App Engine

   Google App Engine allows developers to build and host web applications in cloud-based data centers that Google manages.

2. OpenShift

   OpenShift is an on-premises containerization PaaS software.

The most distinct difference

- IaaS offers administrators more direct control over operating systems
- PaaS offers users greater flexibility and ease of operation.

Example

Let's say a website is wanted to be started. An IaaS product is needed, like Amazon Web Services, to host it and its applications. If a custom feature wanted to be created, a PaaS product like Google App Engine can be used to design it and install it on the site.

IaaS builds the infrastructure of cloud-based technology. PaaS helps developers build custom apps via an API that can be delivered over the cloud.

IaaS can be thought of as the foundation of building a cloud-based service -- whether that's content, software, or the website to sell a physical product, PaaS as the platform on which developers can build apps without having to host them.

6. What is meant by “infrastructure as code”. What are the advantages of this approach?

Infrastructure as code(iAc) is a means by which engineers define the computer systems their code needs to run. Most commonly, these engineers utilize a framework like Chef or Ansible or Puppet to define their infrastructure. Each allows an engineer to define a computer system or network of computer systems necessary to run and support their code.

- **Speed and simplicity**

  IaC allows you to spin up an entire infrastructure architecture by running a script. Not only can you deploy virtual servers, but you can also launch pre-configured databases, network infrastructure, storage systems, load balancers, and any other cloud service that you may need. You can do this quickly and easily for development, staging, and production environments, which can make your software development process much more efficient (more about this later).

- **Configuration consistency**

  IaC completely standardizes the setup of infrastructure so there is reduced possibility of any errors or deviations. This will decrease the chances of any incompatibility issues with your infrastructure and help your applications run more smoothly

- **Minimization of risk**

  Because code can be version-controlled, IaC allows every change to your server configuration to be documented, logged, and tracked. And these configurations can be tested, just like code. So if there is an issue with the new setup configuration, it can be pinpointed and corrected much more easily, minimizing risk of issues or failure.

- **Increased efficiency in software development**

  Infrastructure as Code allows your company to use Continuous Integration and Continuous Deployment techniques while minimizing the introduction of human errors after the development stage.

  Cloud architectures can be easily deployed in multiple stages to make the software development life cycle much more efficient.

- **Cost savings**

  Automating the infrastructure deployment process allows engineers to spend less time performing manual work, and more time executing higher-value tasks. Because of this increased productivity, your company can save money on hiring costs and engineers' salaries.

7. What is meant by "staging" and "production" environments?

Stage

The stage environment is as similar to the production environment as it can be. You'll have all of the code on a server this time instead of a local machine. It'll connect to as many services as it can without touching the production environment. All of the hard core testing happens here. Any database migrations will be tested here and so will any configuration changes. When you have to do major version updates, the stage environment helps you find and fix any issues that come up too.

Production

The production environment is where users access the final code after all of the updates and testing. Once you're in production, any bugs or errors that remain will be found by a user and you can only hope it's something minor

8. Briefly describe at least three metrics that are important to monitor in an application in production.

- **Agile process metrics**

  Agile process metrics focus on how agile teams make decisions and plan. These metrics do not describe the software, but they can be used to improve the software development process.

- **Security metrics**

  Security metrics reflect a measure of software quality. These metrics need to be tracked over time to show how software development teams are developing security responses.

- **Application crash rate (ACR)**

  Application crash rate is calculated by dividing how many times an application fails (F) by how many times it is used (U).

  ACR = F/U

- **Efficiency**

  Efficiency attempts to measure the amount of productive code contributed by a software developer. The amount of churn shows the lack of productive code. Thus a software developer with a low churn could have highly efficient code

- **Impact**

  Impact measures the effect of any code change on the software development project. A code change that affects multiple files could have more impact than a code change affecting a single file.