CSCI 5409 Cloud Computing – Fall, 2023
Week 7 – Lecture 2 (Oct 20, 2023)

Infrastructure as Code

Dr. Lu Yang
Faculty of Computer Science
Dalhousie University
luyang@dal.ca

Housekeeping and Feedback

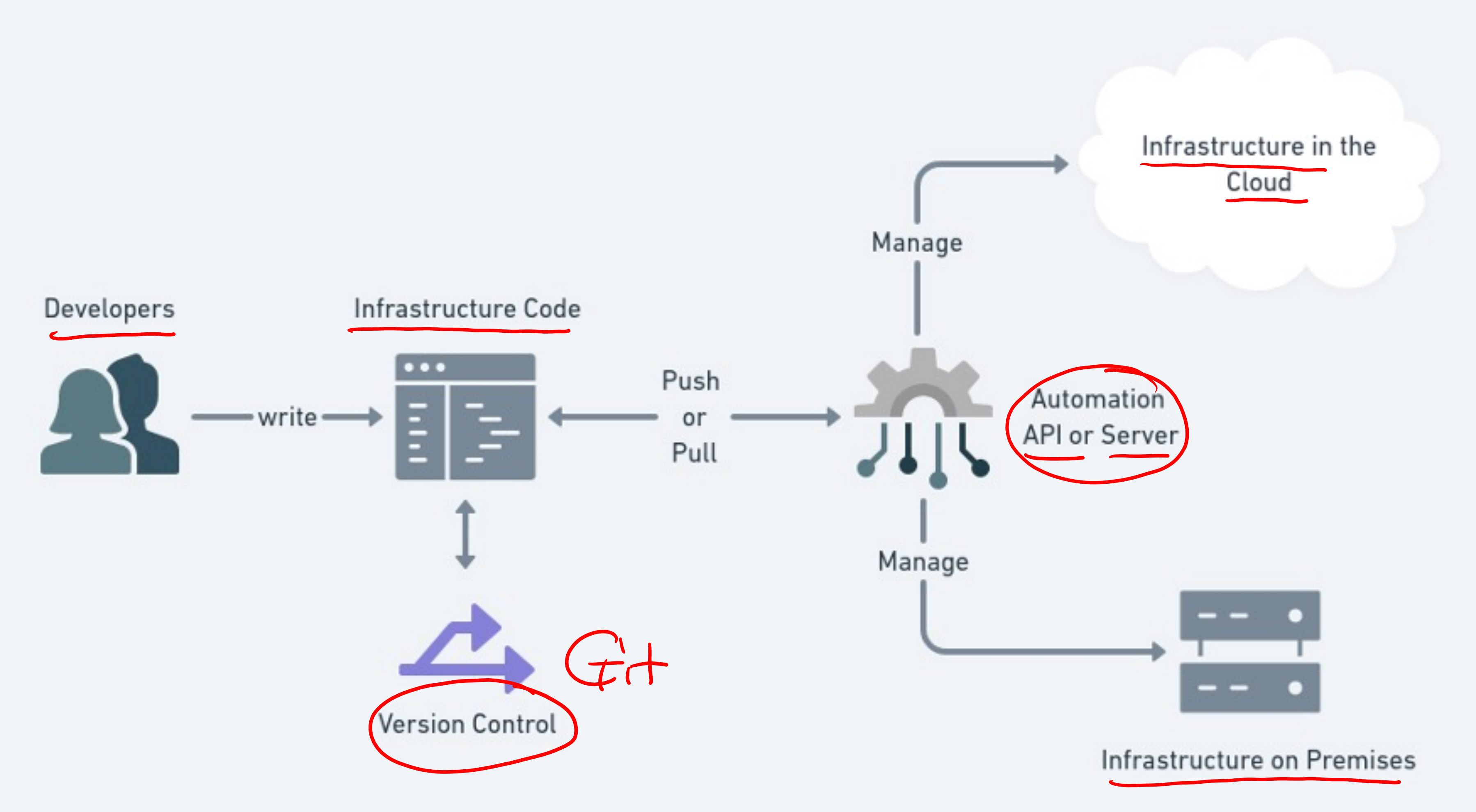
- Start recording
- Midterm happens in class the coming Monday, Oct 23.
- PersistentVolume
  - ReadWriteOnly
  - ReadOnlyMany
  - ReadWriteMany
  - NFS

node

EFS AWS

Filestore GCP

IaC Visualized



[1] https://www.next2it.co.uk/infrastructure-as-code-what-is-it-why-is-it-important/

Benefits of IaC (1/3)

- **Increased speed of deployments**
  - Think of the servers, code and configuration you've done for assignments. How long did it take?
    - Once you've written the template, AWS CloudFormation could reproduce your work in a couple of minutes, in every AWS region worldwide if you wanted.
  - Once your architecture needs scale and automation, manual work is out of the question.
- **Error reduction**
  - Deploying our software is a repetitive task
  - Humans are awesome at some things like debugging and problem solving, but we are terrible at repetitive things because we:
    - are Lazy (skip tests and steps we may consider unimportant)
    - are Imperfect (we forget, we lose focus, get distracted or interrupted)
  - **Handoffs**: The person deploying software may not be the person who wrote the software. They miss subtleties or important steps due to their lack of knowledge. Why not have the developer who writes the software also write the code to deploy the software!

Benefits of IaC (2/3)

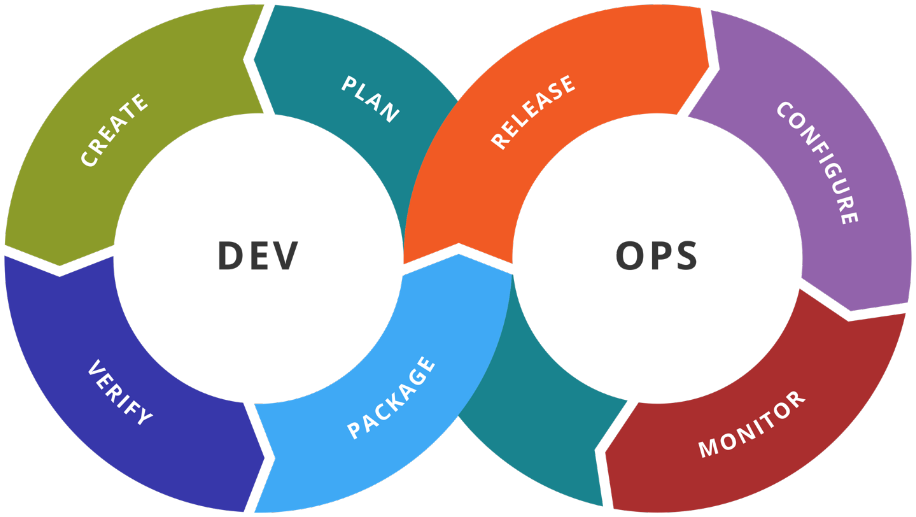
- **Improved infrastructure consistency:**
  - An essential part of your workflow if you are using **geographical diversity** (i.e. reproducing your infrastructure in other regions).
  - Most would argue an essential element to **disaster recovery plans**:
    - How can you guarantee your backups will work if perhaps the only person who knows which button to click might be on vacation?
    - Facilitates testing of disaster recovery plans
    - Allows developers to include the architecture itself as part of the backup, not just the data the architecture manipulates
- **Elimination of configuration "drift"**
  - **Drift:** When our architectures are complicated with many servers, over time with manual configuration each server will diverge from other servers. Configuration changes, OS or dependency updates, and other changes to servers will be done on some servers but not all.
  - Once servers drift apart strange intermittent errors appear.

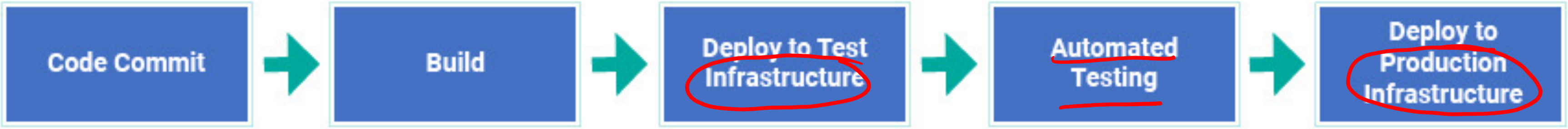
Benefits of IaC (3/3)

- **Cost reduction:**
  - Mistakes cost money, customers can be lost, error in provisioning can produce large cloud provider bills, and always time is lost to find and fix the error.
  - Employees being blocked waiting for a person with cloud admin privileges to provision something to do their work wastes time and money, hindering agility (especially problematic with remote work and different time zones)
- **Self-service options** capable of actions normally only accessible to administrators for efficient development / experimentation
- **Version control** gives us the history of our infrastructure, we can analyze where problems may have been introduced, or revert to stable ground if an experiment fails
- **Improved "Whole Team" understanding**: all developers understand the IT resource architecture in the cloud, which helps them to make better implementation choices

IaC in CI/CD (1/3)

- Automation removes human error, reduces handoffs
- Experimentation is made even more possible through time savings and controlled access to infrastructure
- Continual learning/experimentation are required to stay aware of the latest tools providing competitive advantage, ignored at your company's peril!

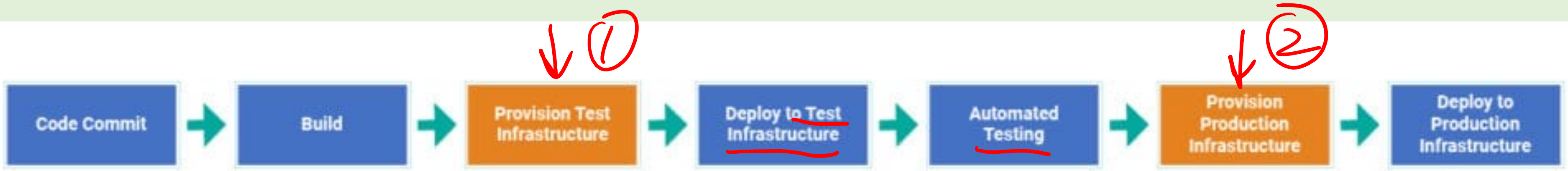



Code Commit → Build → Deploy to Test Infrastructure → Automated Testing → Deploy to Production Infrastructure

https://en.wikipedia.org/wiki/DevOps_toolchain#/media/File:Devops-toolchain.svg
https://www.logicworks.com/blog/2020/10/cicd-iac-pipeline-part-1/

IaC in CI/CD (2/3)

- Cloud-native teams want to control the delivery of this cloud infrastructure using the same CI/CD pipelines that control the delivery of their software applications.
- Cloud-native software development teams code steps in their application's CI/CD pipeline to dynamically provision the needed IaC definition and then deploy the desired version of their application onto this infrastructure.
- The CI/CD pipeline can easily be configured to launch cloned development or staging environments.
- This combined IaC and CI/CD pipeline process provisions cloud infrastructure just-in-time, thus optimizing cloud costs. The cloud-native team can now rapidly and reliably develop, test, and deploy enhancements to both their application and its cloud infrastructure.



Code Commit → Build → Provision Test Infrastructure → Deploy to Test Infrastructure → Automated Testing → Provision Production Infrastructure → Deploy to Production Infrastructure

https://www.logicworks.com/blog/2020/10/cicd-iac-pipeline-part-1/

IaC in CI/CD (3/3)

How to Enable IaC and CI/CD?

- **Step 1**: Define Infrastructure as Code

  Puppet

  The first step is to define infrastructure as code using tools like Terraform, Ansible, or Chef. With IaC, infrastructure can be defined in a text file, which can be version controlled and deployed automatically.

- **Step 2**: Create a CI/CD Pipeline

  The second step is to create a CI/CD pipeline using tools like Jenkins, GitLab CI/CD, or Azure DevOps. With a CI/CD pipeline, code changes can be automatically tested, built, and deployed to production.

- **Step 3**: Integrate IaC and CI/CD

  The final step is to integrate IaC and CI/CD. This can be done by deploying infrastructure changes automatically as part of the CI/CD pipeline. This ensures that the infrastructure is consistent and up-to-date with the latest code changes.

https://www.linkedin.com/pulse/enabling-infrastructure-code-iac-cicd-key-benefits-customers-bilal/

Infrastructure from Code (IfC)

- While IaC tools are constantly improving and adding higher-level features, they all fundamentally require a **human** to declare the specific infrastructure components they need, usually in a **fair amount of detail**.
  - This means that a developer needs to understand not just what resources their application needs, but the permission models, dependent resources, and communication links between those resources.

- Infrastructure-from-Code (IfC) is a new way of thinking about cloud infrastructure, and represents the next step in a line of innovations that makes spinning up infrastructure easier and more seamless for developers.
- Infrastructure-from-Code (IfC) is a process that analyzes your application code to infer the cloud resources you need, and then creates and maintains them without you having to manually define them.
- There are a few different approaches to IfC, with different providers taking different bets on how it'll look.
  https://klo.dev/state-of-infrastructure-from-code-2023/

IaC Demo

- https://www.youtube.com/watch?v=-Ep44eHUpxc