Carnegie Mellon

institute for SOFTWARE RESEARCH

DevOps-Continuous Integration

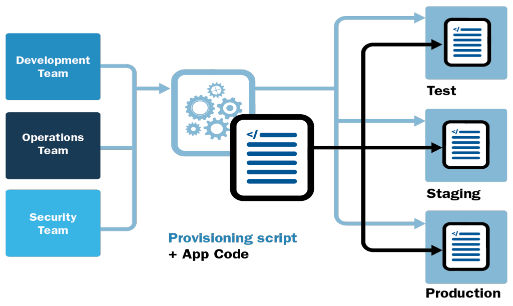
Infrastructure as code

1

Carnegie Mellon University

Infrastructure as Code (IaC)

A program that creates infrastructure



institute for SOFTWARE RESEARCH

2

Carnegie Mellon University

Outline

- **Basic concepts**
- Environments
- Vendor lock-in

© Len Bass 2021 3

institute for SOFTWARE RESEARCH

3

Carnegie Mellon University

What is IaC?

- Infrastructure as Code (IaC) is the management of infrastructure (networks, virtual machines, load balancers, and connection topology) using code segments in various languages. E.g.
  - Command line scripting
  - Provisioning specifications, e.g. Vagrant, Cloud Formation, Chef, Puppet, Ansible
  - Specification files for various tools, e.g Dockerfile

© Len Bass 2021 4

institute for SOFTWARE RESEARCH

4

Uniformity

- Uniformity in a variety of contexts is important
- Teams should have uniform development environments
- Various stages of the release pipeline should have environments that are as uniform as possible.
- IaC is a means for keeping environments consistent and uniform.

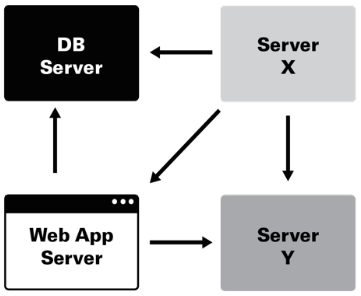
"Snowflakes"

- Without IaC, teams must maintain the settings of individual deployment environments.
- Over time, each environment becomes a *snowflake*, that is, a unique configuration that cannot be reproduced automatically.

Carnegie Mellon University

Why do we see snowflakes?

- Suppose a system requires multiple servers networked together



Each server is to be provisioned uniquely, providing a specific service

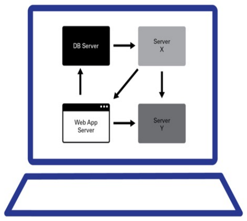
© Len Bass 2021

isr institute for SOFTWARE RESEARCH

Carnegie Mellon University

Snowflake scenario - 1

- At the time of development, developers write code assuming this specific infrastructure configuration
- During development, the environment is created by creating and provisioning virtual machines or simply running the component services on a developer's computer



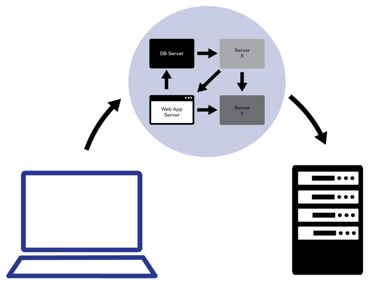
© Len Bass 2021

isr institute for SOFTWARE RESEARCH

Carnegie Mellon University

Snowflake scenario - 2

- At deployment, the deployment environment (Test / QA / Production) is manually provisioned to match the expected infrastructure configuration



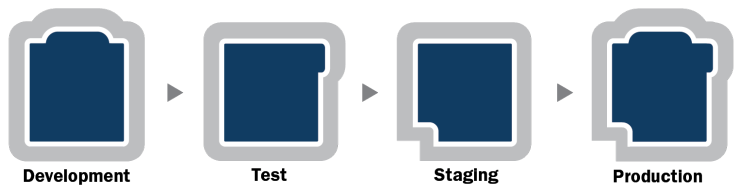
This often means manual installation of packages, configurations, and networking components…

…and room for human error

institute for SOFTWARE RESEARCH

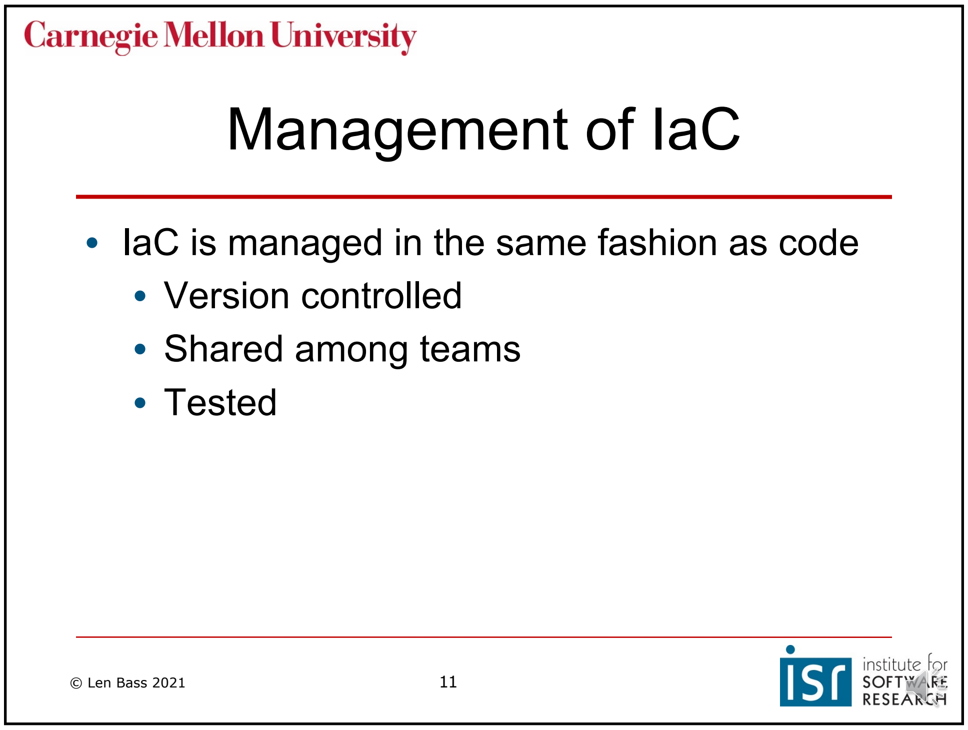
Carnegie Mellon University

Divergence



- Environments are independent, volatile, and easily manipulated.
- Without care, they will diverge and create snowflakes.
- Hence, IaC

institute for SOFTWARE RESEARCH



Carnegie Mellon University

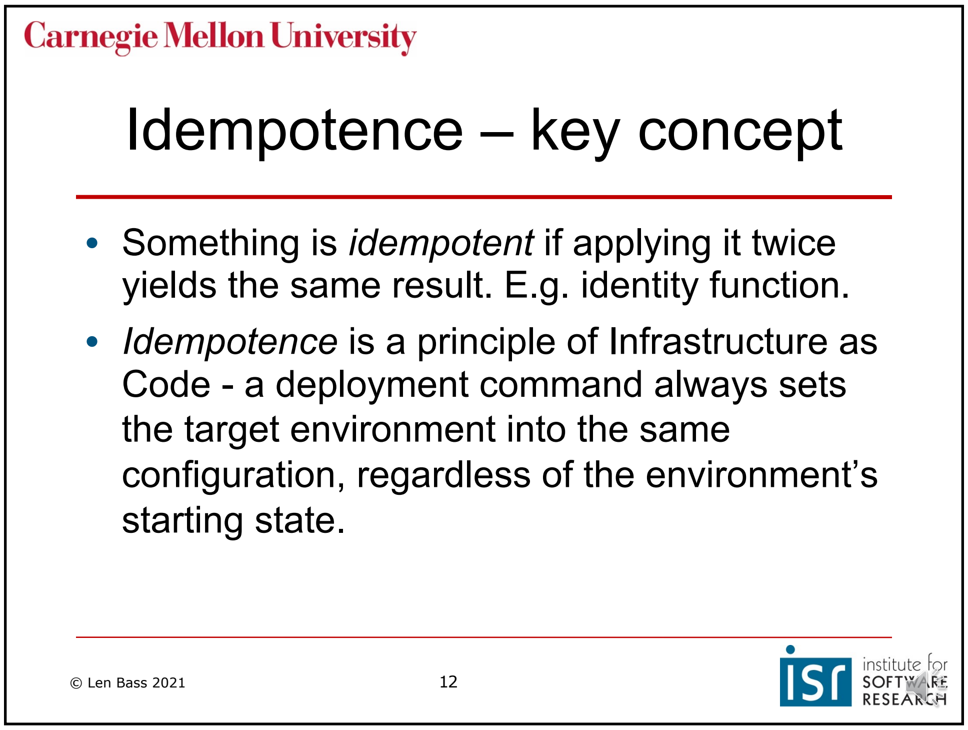
Management of IaC

- IaC is managed in the same fashion as code
  - Version controlled
  - Shared among teams
  - Tested

© Len Bass 2021 11

institute for SOFTWARE RESEARCH

11



Carnegie Mellon University

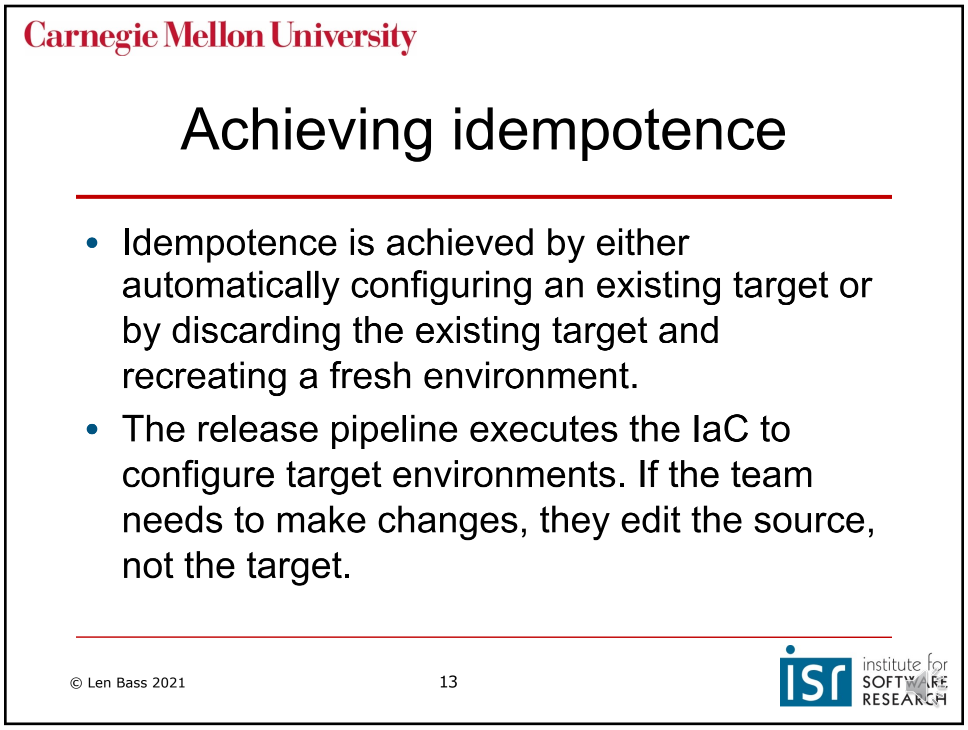
Idempotence – key concept

- Something is *idempotent* if applying it twice yields the same result. E.g. identity function.
- *Idempotence* is a principle of Infrastructure as Code - a deployment command always sets the target environment into the same configuration, regardless of the environment’s starting state.

© Len Bass 2021 12

institute for SOFTWARE RESEARCH

12



Carnegie Mellon University

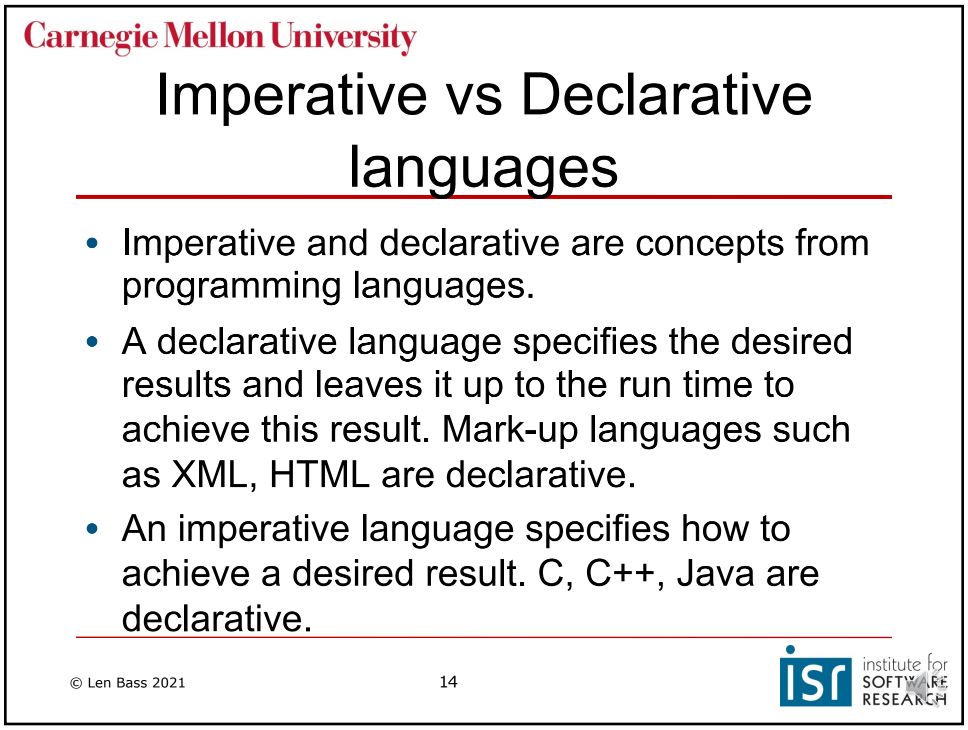
Achieving idempotence

- Idempotence is achieved by either automatically configuring an existing target or by discarding the existing target and recreating a fresh environment.
- The release pipeline executes the IaC to configure target environments. If the team needs to make changes, they edit the source, not the target.

© Len Bass 2021 13

institute for SOFTWARE RESEARCH

13



Carnegie Mellon University

Imperative vs Declarative languages

- Imperative and declarative are concepts from programming languages.
- A declarative language specifies the desired results and leaves it up to the run time to achieve this result. Mark-up languages such as XML, HTML are declarative.
- An imperative language specifies how to achieve a desired result. C, C++, Java are declarative.

© Len Bass 2021 14

institute for SOFTWARE RESEARCH

14

Carnegie Mellon University

IaC languages

- IaC languages are typically declarative although they may have some imperative portions.
- Command line scripting is declarative but may invoke an imperative procedure.
- Declarative languages make achieving idempotence easier.

15 institute for SOFTWARE RESEARCH

15

Carnegie Mellon University

Outline

- Basic concepts
- **Environments**
- Vendor lock-in

16 institute for SOFTWARE RESEARCH

16

Carnegie Mellon University

Environments

- If IaC manages environments – what is an environment?
- An *environment* is a collection of provisioned resources.
- These days, typically a collection of resources in the cloud. Although it could be resources on a developer's computer.

© Len Bass 2021 17

17

Carnegie Mellon University

What kinds of environments?

- Environments exist for
  - Development
  - Integration
  - Testing
  - Staging
  - Production

© Len Bass 2021 18

18

Carnegie Mellon University

Environments have a life cycle

1. Creation
2. Provisioning
3. Use
4. Saving
5. Tear Down
   - Creation, provisioning, saving, and tear down can all be managed by IaC
   - Use is typically not managed by IaC.

Carnegie Mellon University

Creating an environment

- Cloud providers have specialized tools to create an environment.
  - AWS has CloudFormation
  - Azure has ARM (Azure Resource Manager)
- These tools
  - Allocate VMs of various types
  - Specify access rules

Carnegie Mellon University

Access rules

- An access rule specifies the relationship between an identity (class of users and systems) and resource permissions (what can a member of that class do with that resource).
- Users must authenticate (prove who they are) and be authorized (have the right to do particular operations on the resource)

© Len Bass 2021 21

isr institute for SOFTWARE RESEARCH

21

Carnegie Mellon University

Cloud providers provide templates

- The constrction of an IaC program is simplified by the availability of templates provided by cloud providers.

© Len Bass 2021 22

isr institute for SOFTWARE RESEARCH

22

Carnegie Mellon University

Sample Cloud Formation template (with no access control)

```
"AWSTemplateFormatVersion" : "2010-09-09",
"Description" : "A sample template", "Resources" : {
"MyEC2Instance" : { "Type" : "AWS::EC2::Instance",
"Properties" : { "ImageId" : "ami-2f726546",
"InstanceType" : "t1.micro", "KeyName" : "testkey",
"BlockDeviceMappings" : [ { "DeviceName" :
"/dev/sdm", "Ebs" : { "VolumeType" : "io1", "Iops" :
"200", "DeleteOnTermination" : "false", "VolumeSize" :
"20" } } ] } } } }
```

institute for SOFTWARE RESEARCH

Carnegie Mellon University

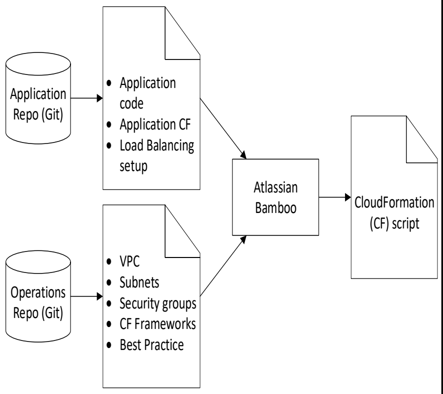
Vulnerabilities

- However, inappropriate access control is a source of many production vulnerabilities.
- One study by Palo Alto Networks found over 200,000 vulnerabilities in CloudFormation specifications due to inadequate access controls.
- Another study cited by Accurics found that 93% of specifications had misconfigured cloud storage access controls.

institute for SOFTWARE RESEARCH

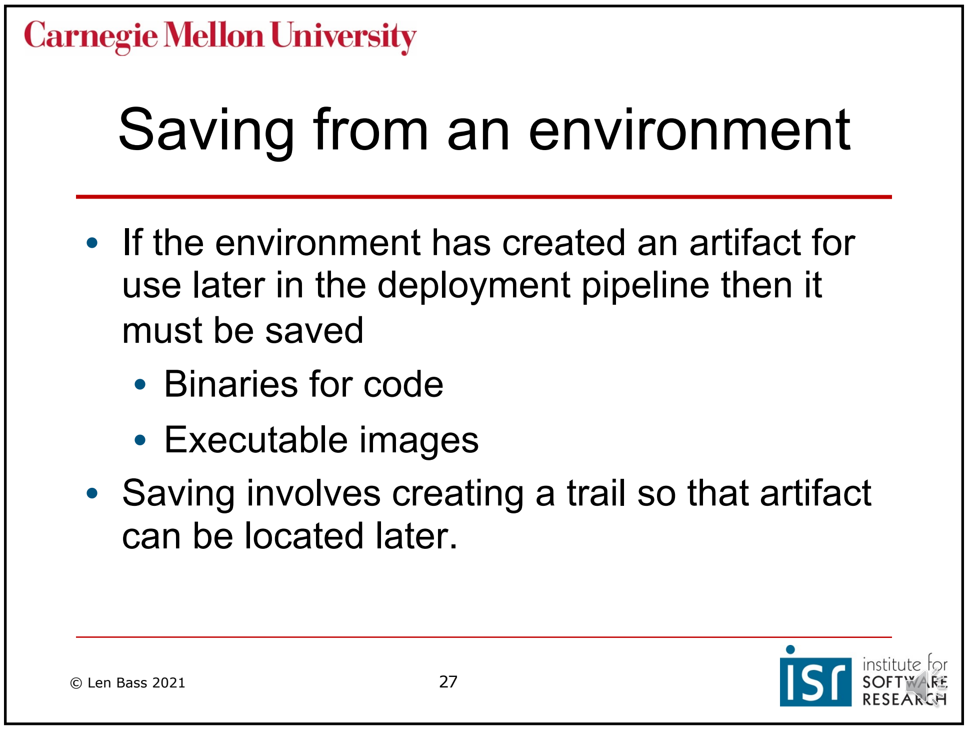
Separating concerns

- Use Cloud Formation (CF) scripts
- App development group maintains app specific CF scripts
- Operations group maintains global CF scripts for access control and other operations concerns
- Merge into a single CF script



Provisioning an environment

- Load created VMs with application specific software
- Done using provisioning tools. E.g.
  - Vagrant
  - CloudFormation has provisioning capabilities
  - Chef, Puppet, Ansible
  - …



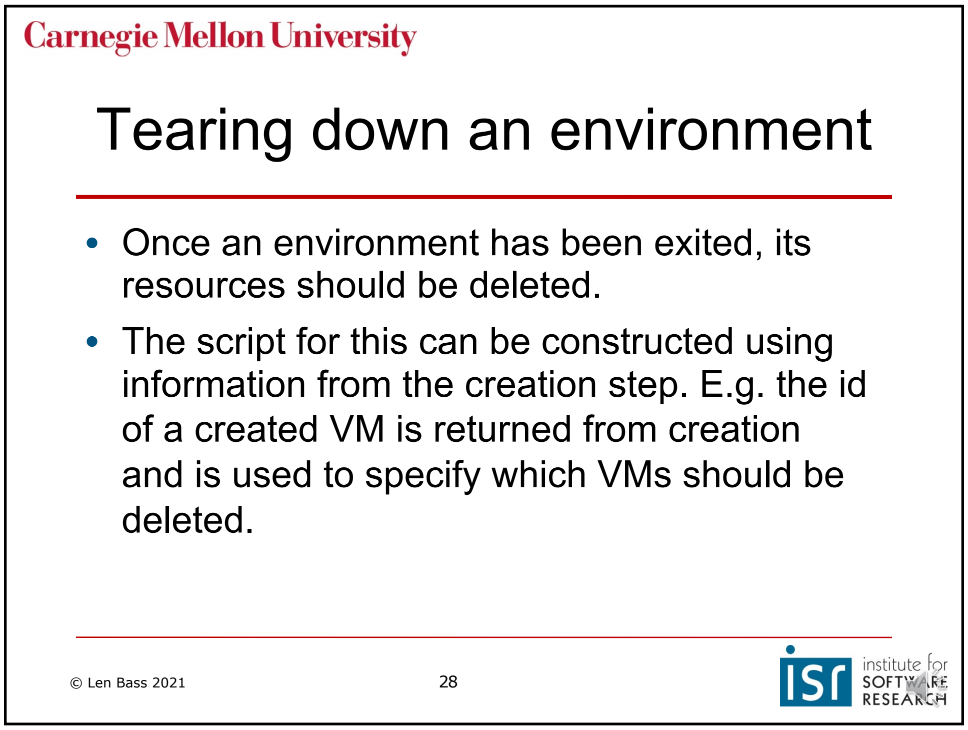
Carnegie Mellon University

Saving from an environment

- If the environment has created an artifact for use later in the deployment pipeline then it must be saved
  - Binaries for code
  - Executable images
- Saving involves creating a trail so that artifact can be located later.

isr institute for SOFTWARE RESEARCH

27



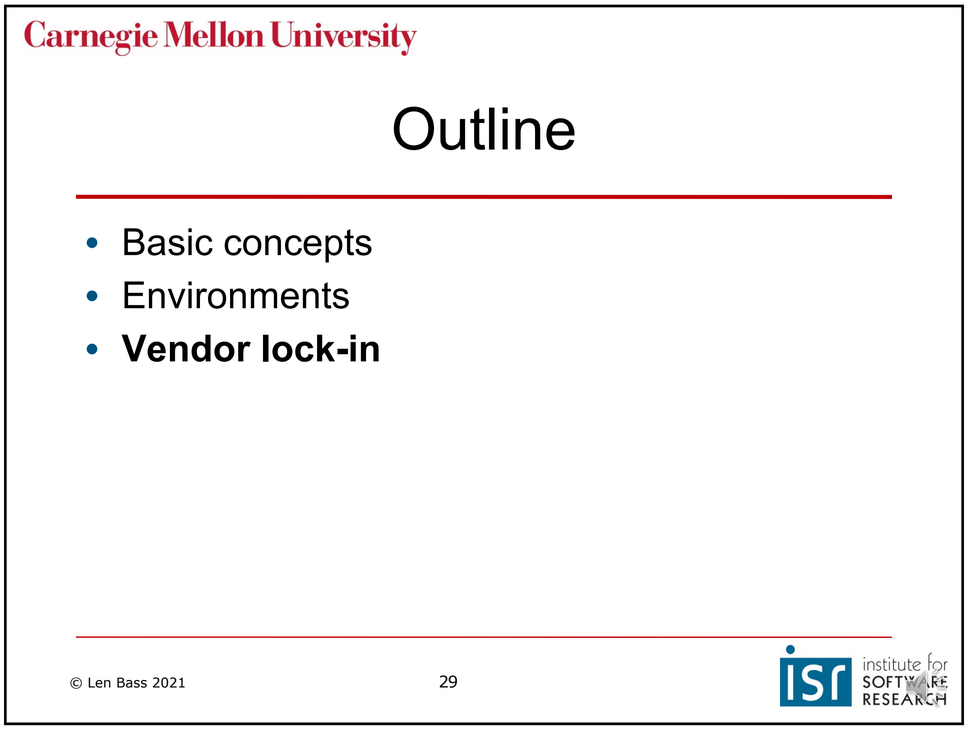
Carnegie Mellon University

Tearing down an environment

- Once an environment has been exited, its resources should be deleted.
- The script for this can be constructed using information from the creation step. E.g. the id of a created VM is returned from creation and is used to specify which VMs should be deleted.

isr institute for SOFTWARE RESEARCH

28



Carnegie Mellon University

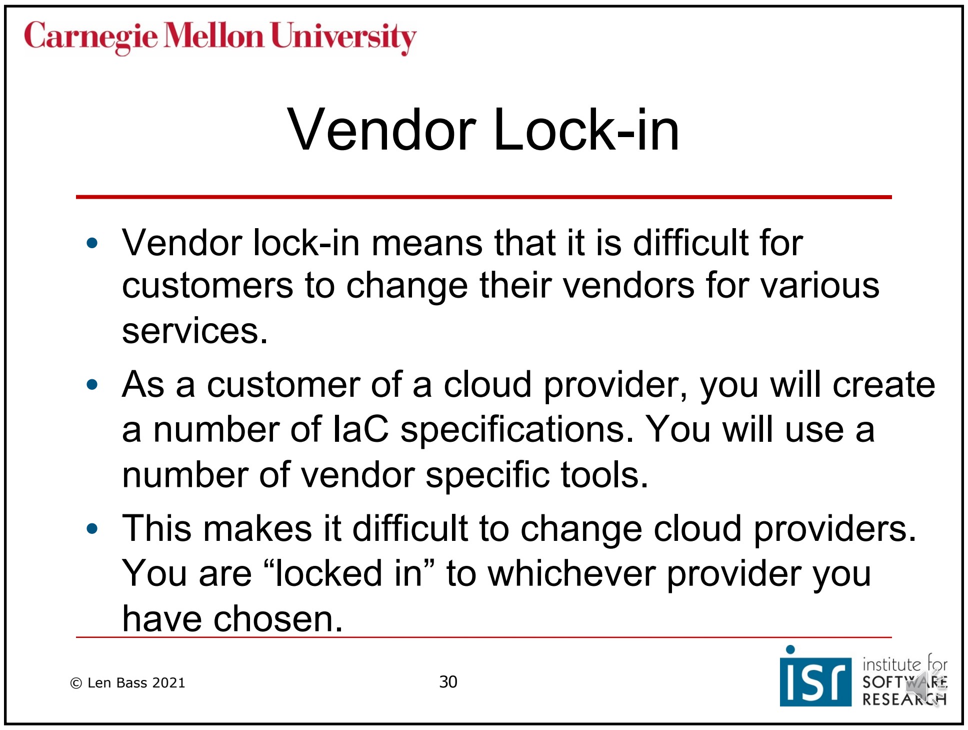
Outline

- Basic concepts
- Environments
- **Vendor lock-in**

© Len Bass 2021 29

institute for SOFTWARE RESEARCH

29



Carnegie Mellon University

Vendor Lock-in

- Vendor lock-in means that it is difficult for customers to change their vendors for various services.
- As a customer of a cloud provider, you will create a number of IaC specifications. You will use a number of vendor specific tools.
- This makes it difficult to change cloud providers. You are “locked in” to whichever provider you have chosen.

© Len Bass 2021 30

institute for SOFTWARE RESEARCH

30

Carnegie Mellon University

Techniques to avoid vendor lock-in

- Use tools that work for various vendors
- Use intermediaries to specify access to various resources. Means that if you change vendors, only the intermediaries must be changed.

institute for SOFTWARE RESEARCH

Carnegie Mellon University

Summary

- IaC involves the creation of a number of specifications to manage infrastructure
- These specifications should be managed as code – version controlled and tested
- IaC specifications are used to create and manage environments throughout an environment’s lifecycle.
- Use of IaC may lock you into a specific cloud vendor.

institute for SOFTWARE RESEARCH