
Linux Academy
Amazon Web Services
AWS Developer Tools: Course Introduction

CodeCommit:

- A managed, source code version control service that host private Git based repositories.

CodeDeploy:

- A code deployment automation service to EC2 or on-premise instances.

CodePipeline:

- A continuous delivery service that provides tools to model, visualize, and automate the software release process.

Linux Academy

AWS Developer Tools Course:

Who is this course for?

- Anyone who wants to expand their knowledge of code management and deployment services:
  - A 10 year I.T. veteran
  - A budding or experienced developer
  - Someone who has no idea what code management and deployment services are

Course Prerequisites:

- There are no specific prerequisites for this course.
- However, I will assume that you are somewhat familiar with the following AWS services:
  - S3
  - EC2
  - IAM
- Basic Linux knowledge and hand-on experience


Linux Academy
Amazon Web Services
AWS Developer Tools: Course Introduction
Thank you for watching!


Linux Academy
Amazon Web Services
Developer Tools: CodeCommit

What is CodeCommit?

- AWS CodeCommit is a managed source code control service that host private Git repositories

- Benefits:
  - Highly available, scalable & fault tolerant
  - No size limit
  - Integrates with other AWS services (i.e. CodePipeline, Lambda & SNS)
  - Easily Migrate files from other Git-based repositories
  - Works with existing Git-based tools

Linux Academy

Explain Like I am Five (ELI5):

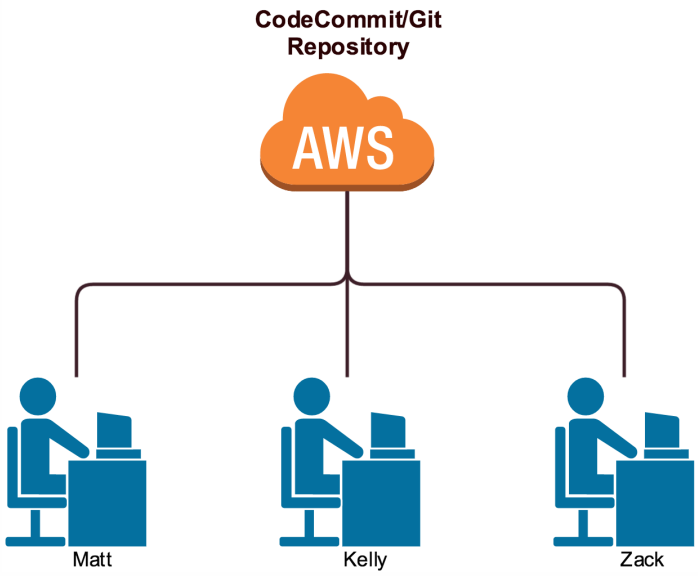
- At the highest level, CodeCommit is a communication tool.
- It is an service that allows developers to collaborate on a project and easily manage, share, update, and coordinate the code they are independently working on.

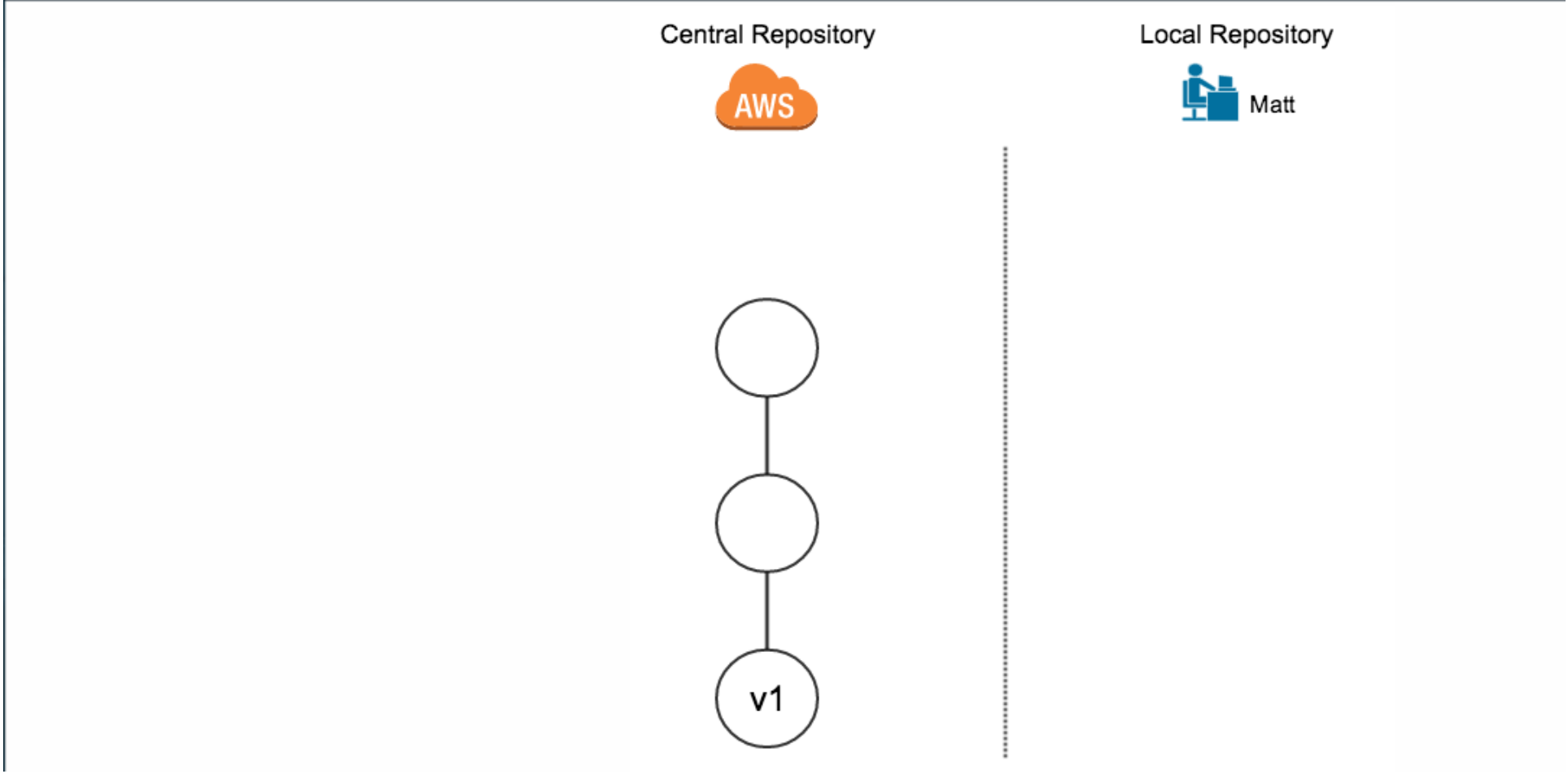
Meet our Imaginary Developers!

- Matt
- Kelly
- Zack

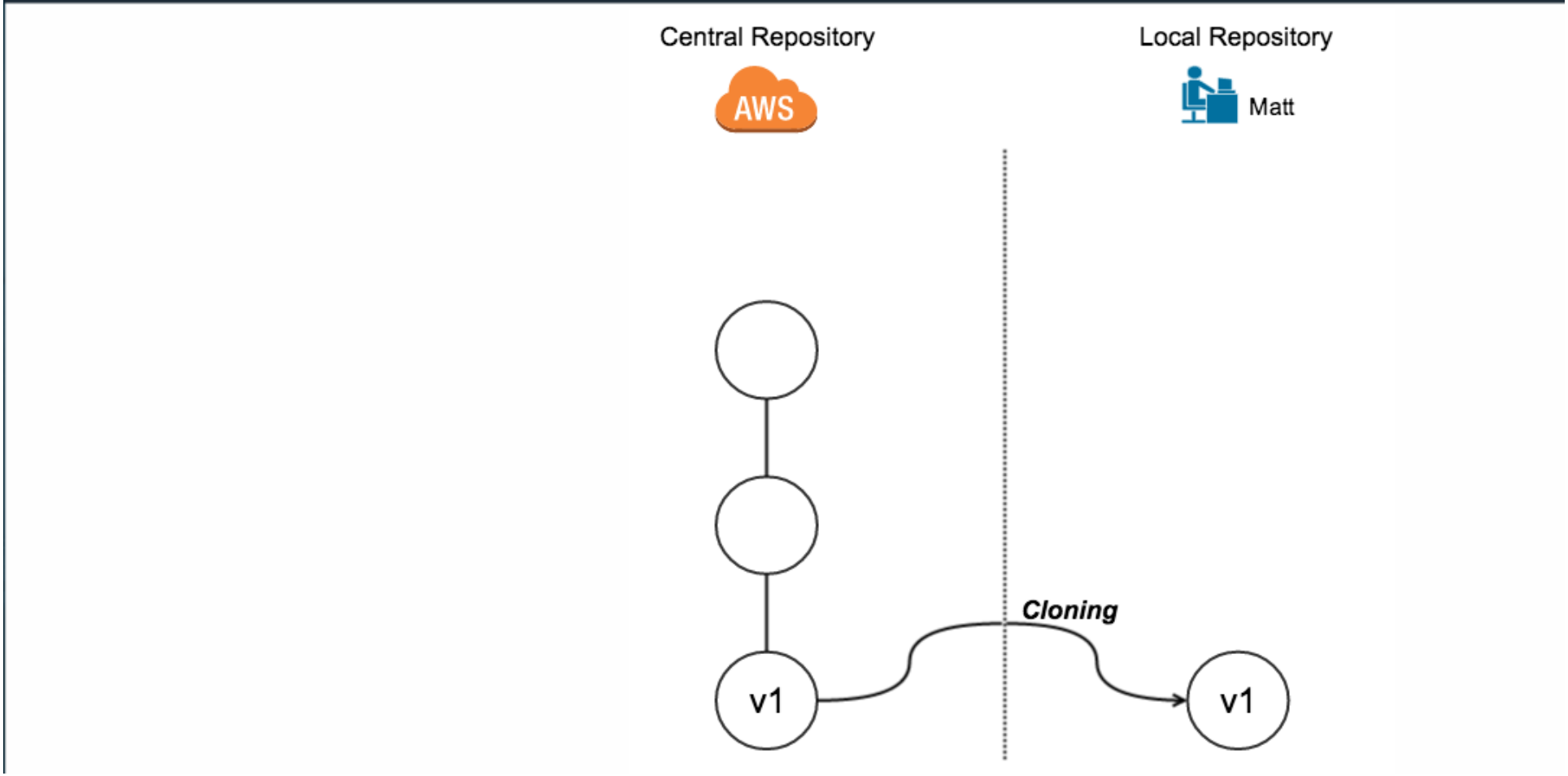
They are working on a project called:

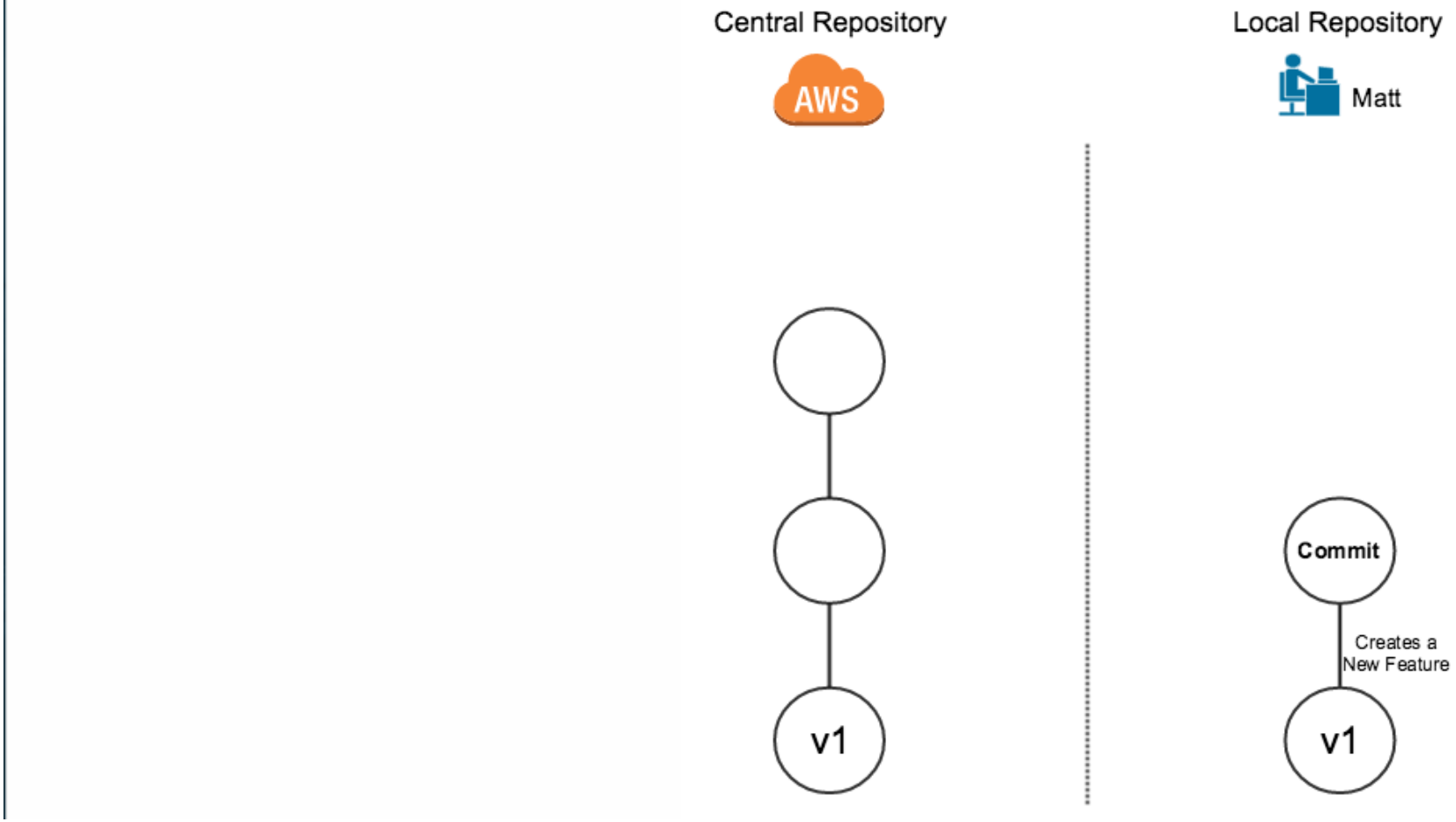
WonderWidgets



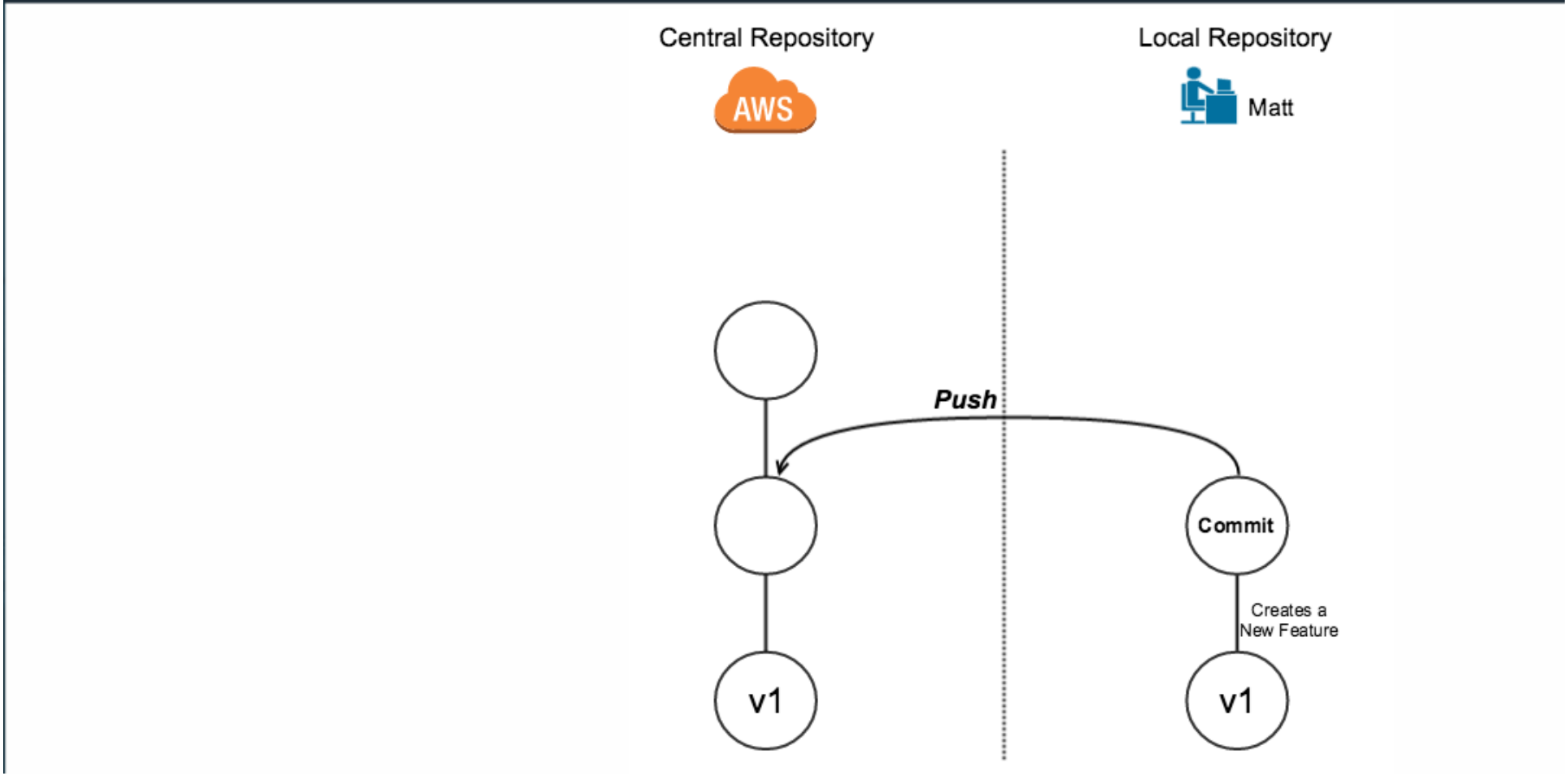

Central Repository
Local Repository
AWS
Matt
v1

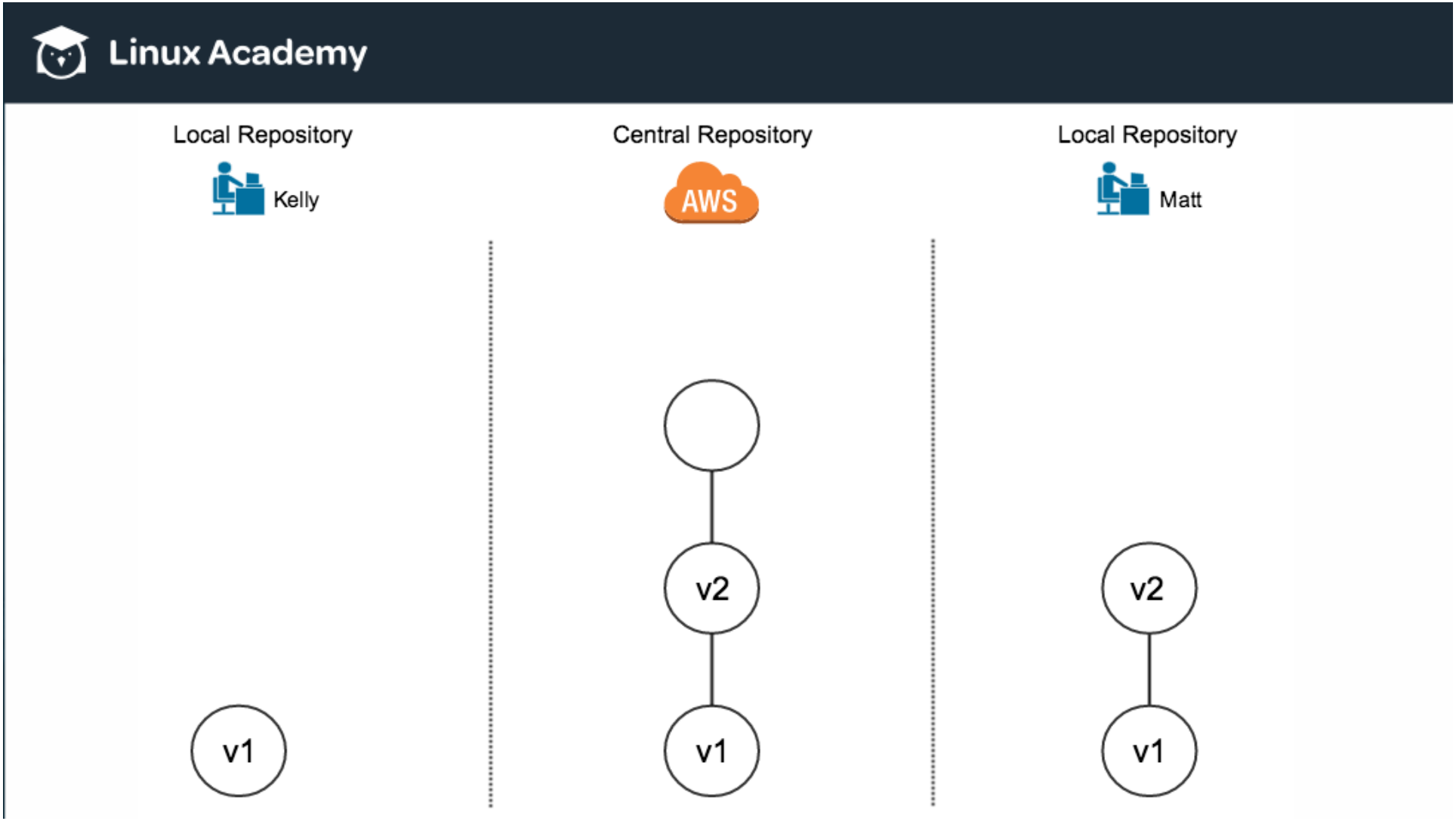
Linux Academy



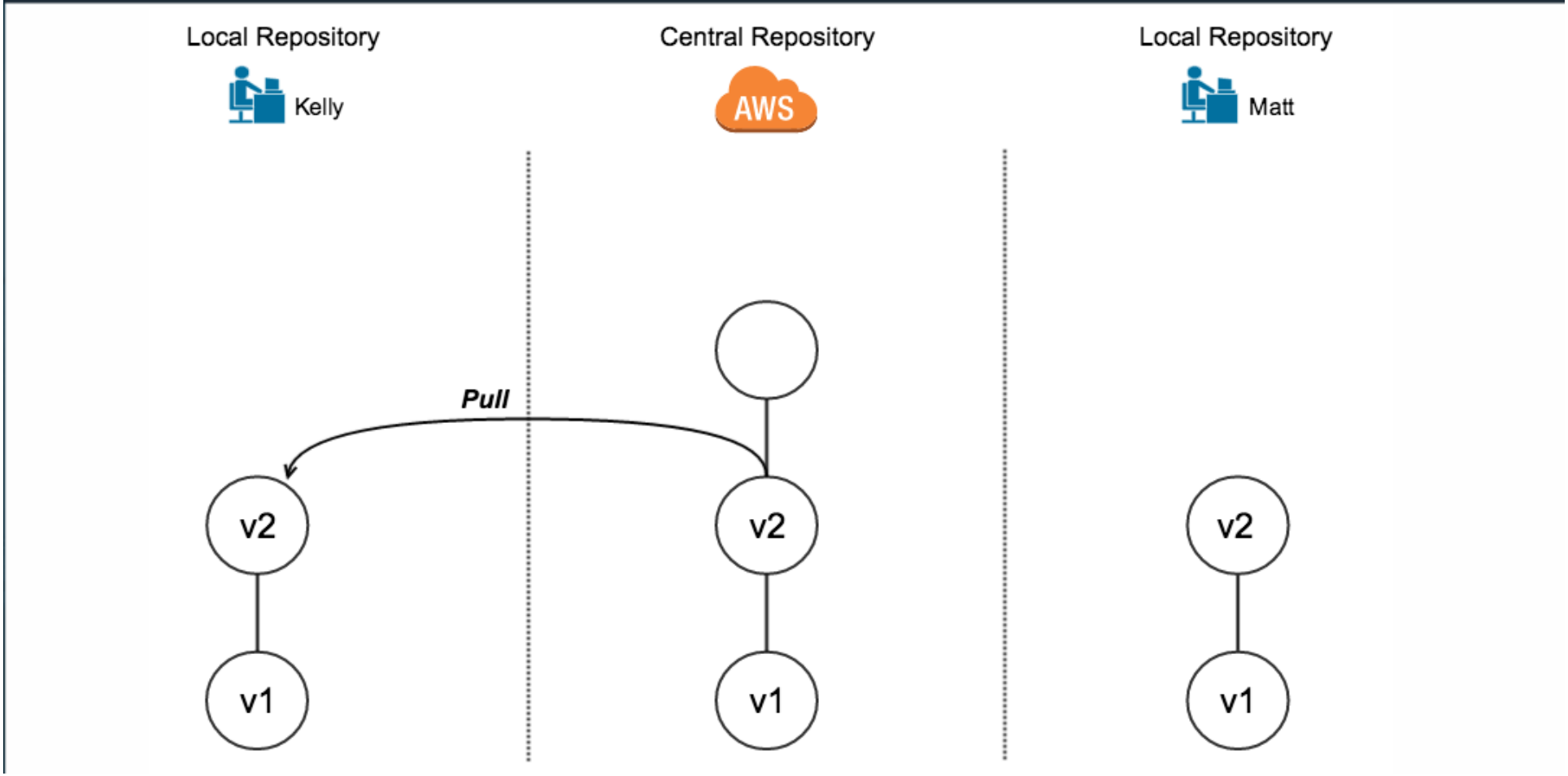

Central Repository
Local Repository
AWS
Matt
v1
Commit
Creates a
New Feature
v1

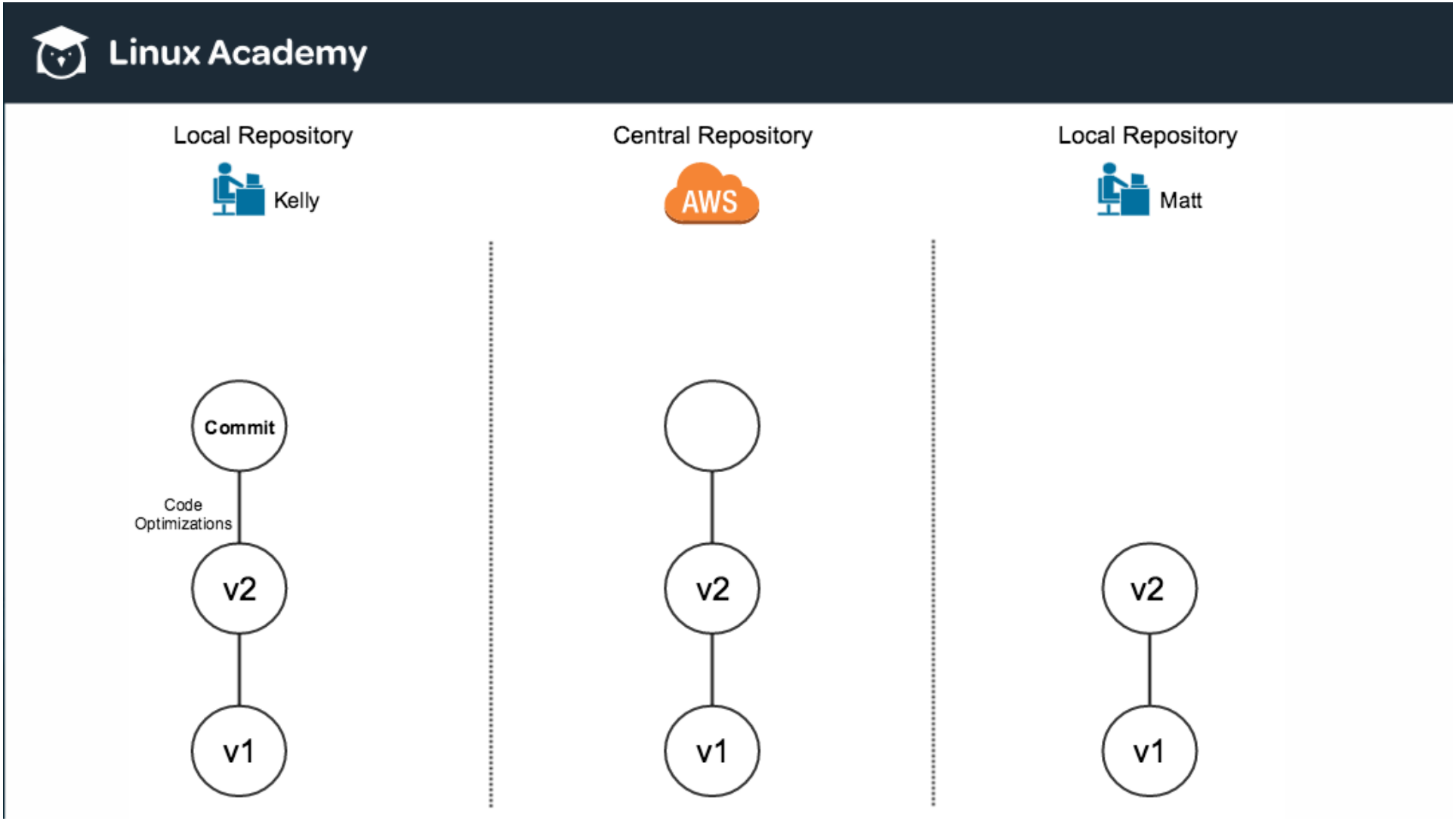
Linux Academy



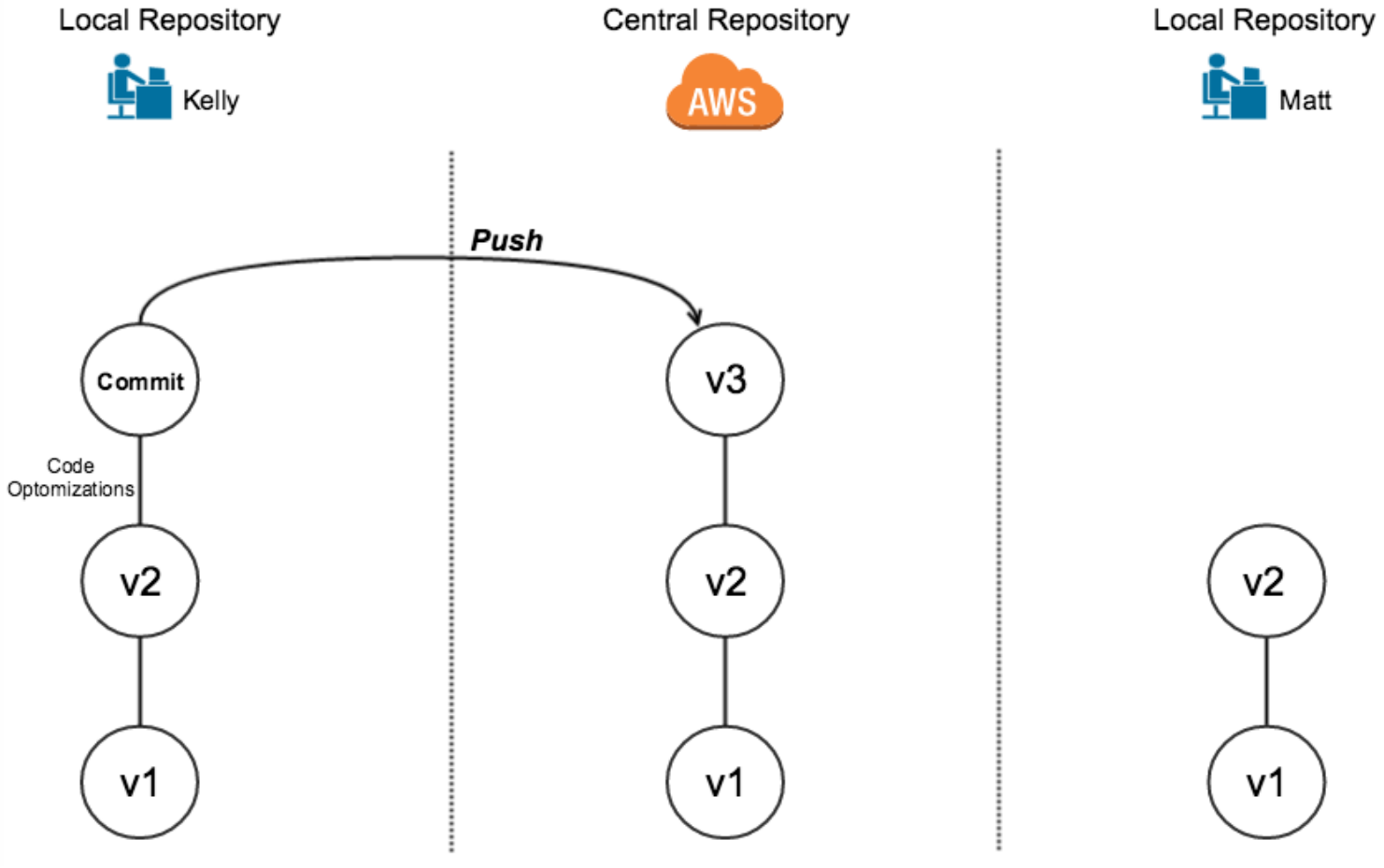

Linux Academy
Local Repository
Kelly
Central Repository
AWS
Local Repository
Matt
v1
v2
v1
v2
v1

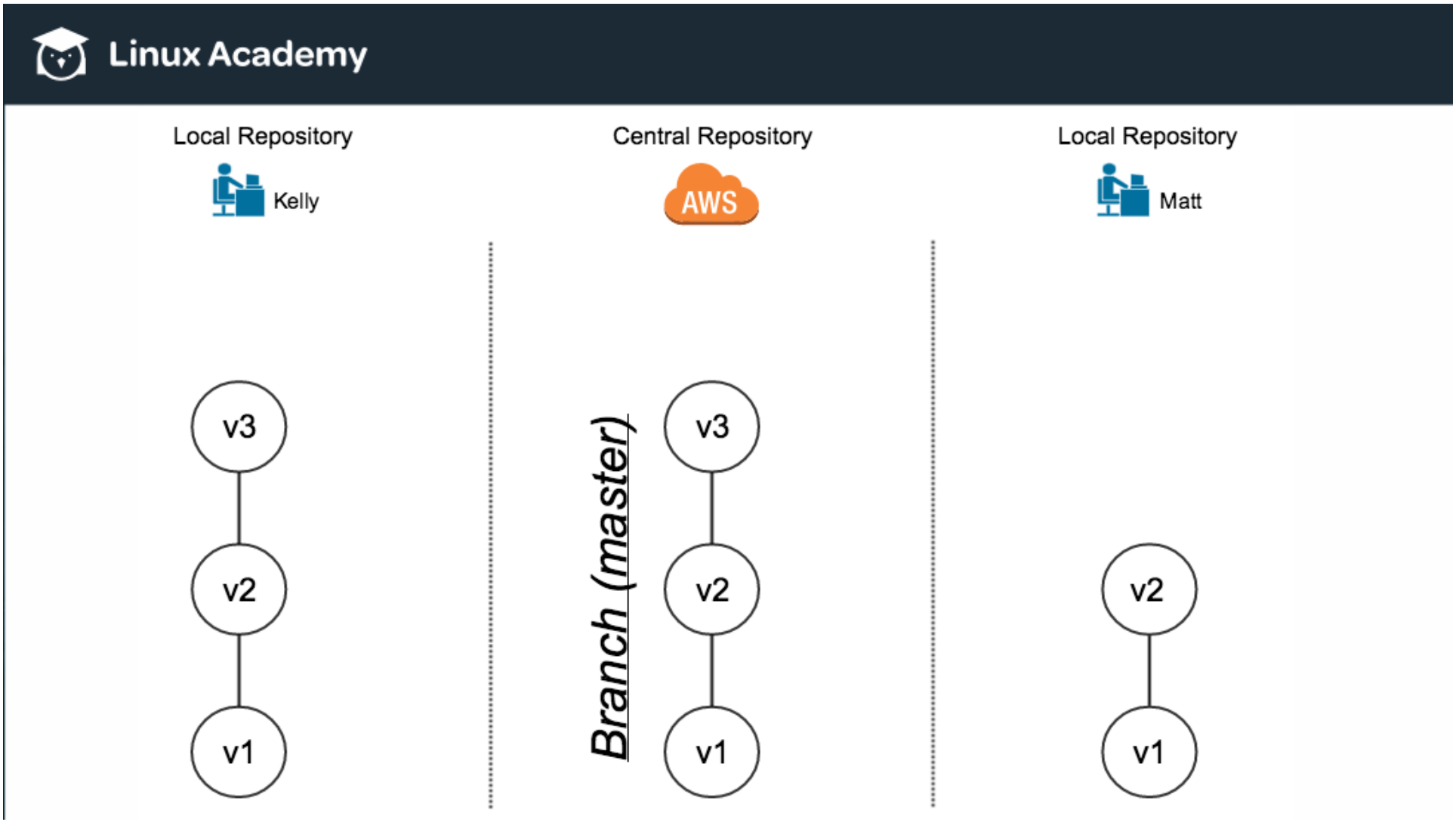
Linux Academy

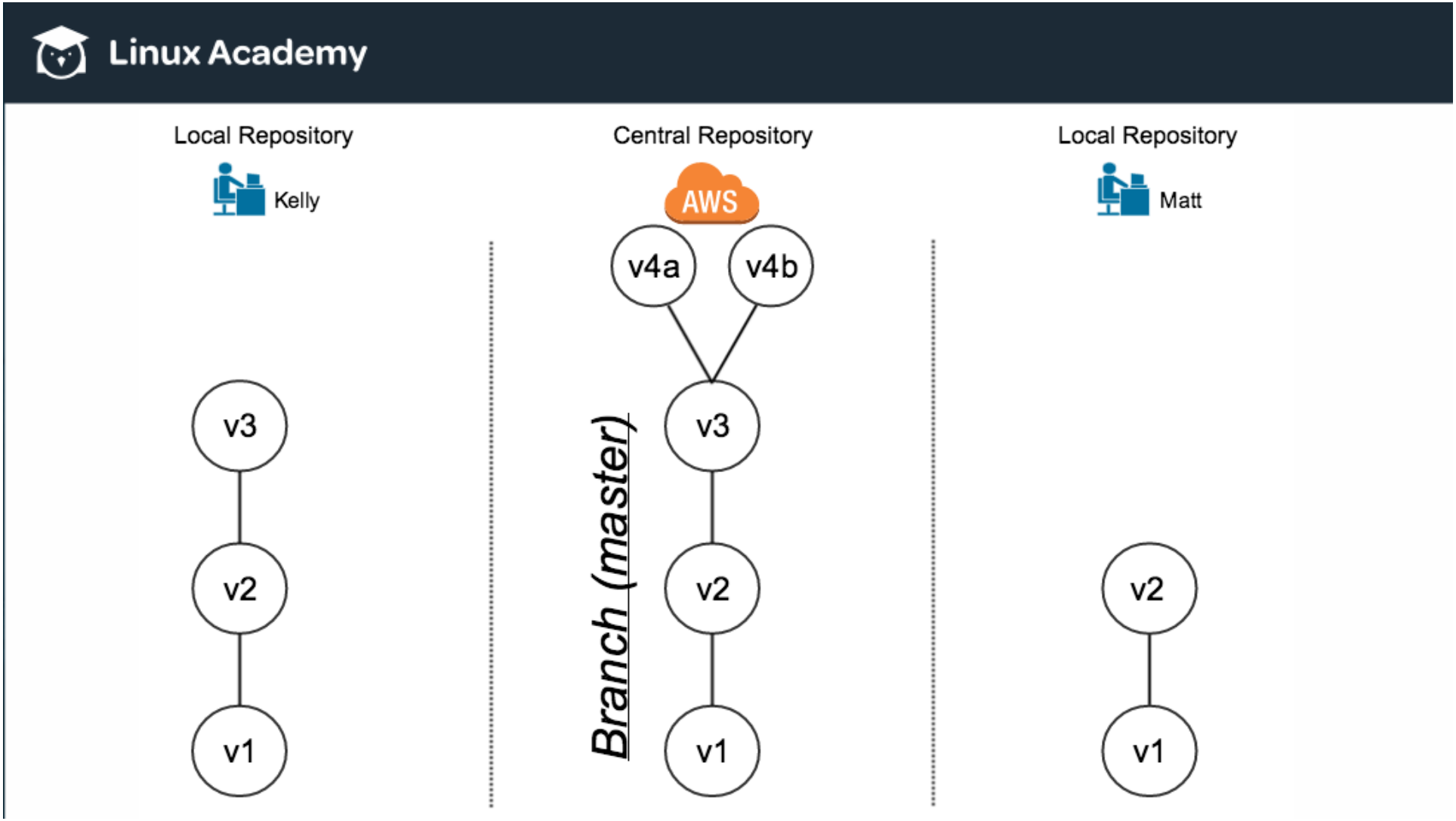


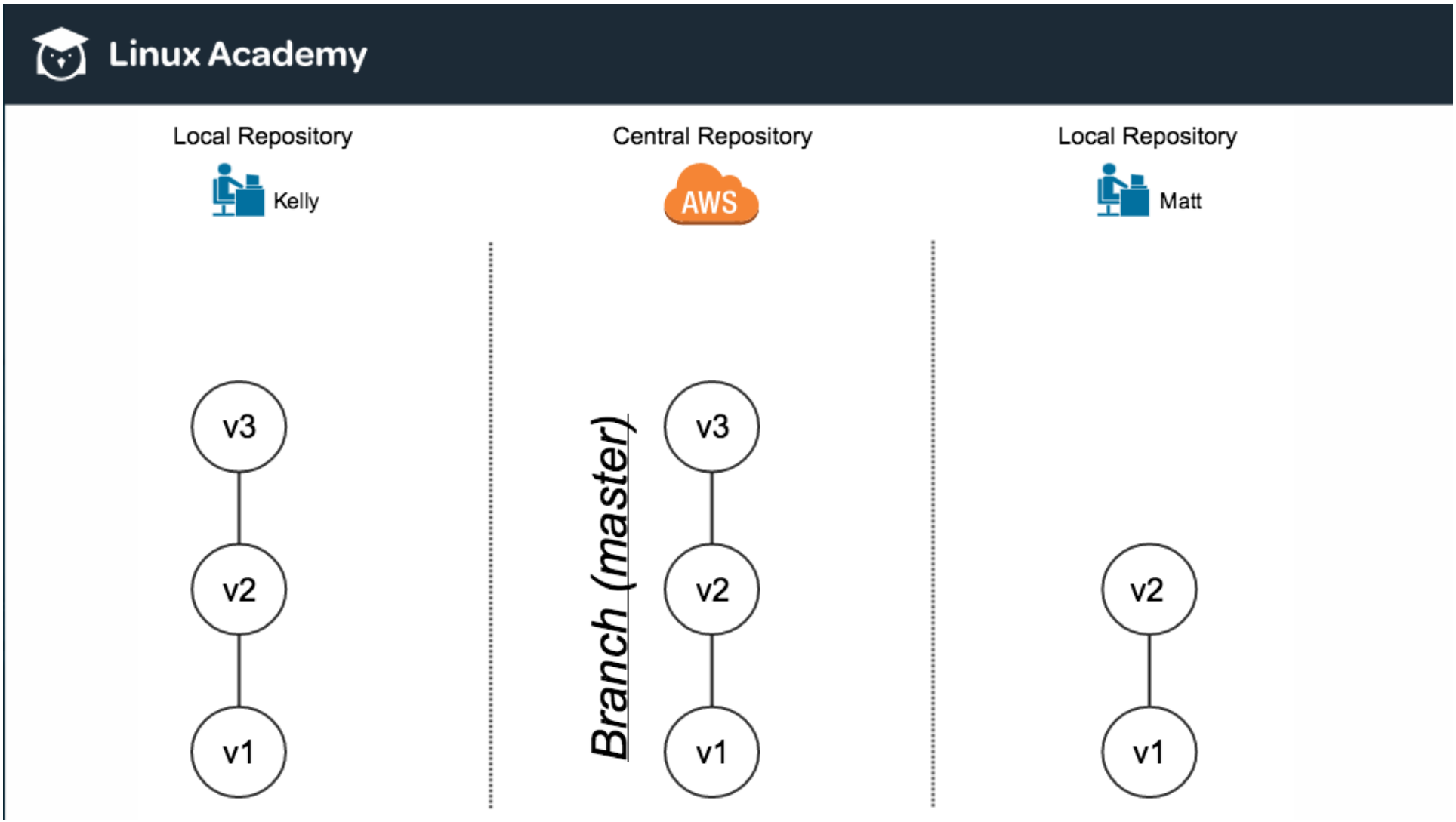

Linux Academy
Local Repository
Kelly
Central Repository
AWS
Local Repository
Matt
Commit
Code
Optimizations
v2
v1
v2
v1
v2
v1

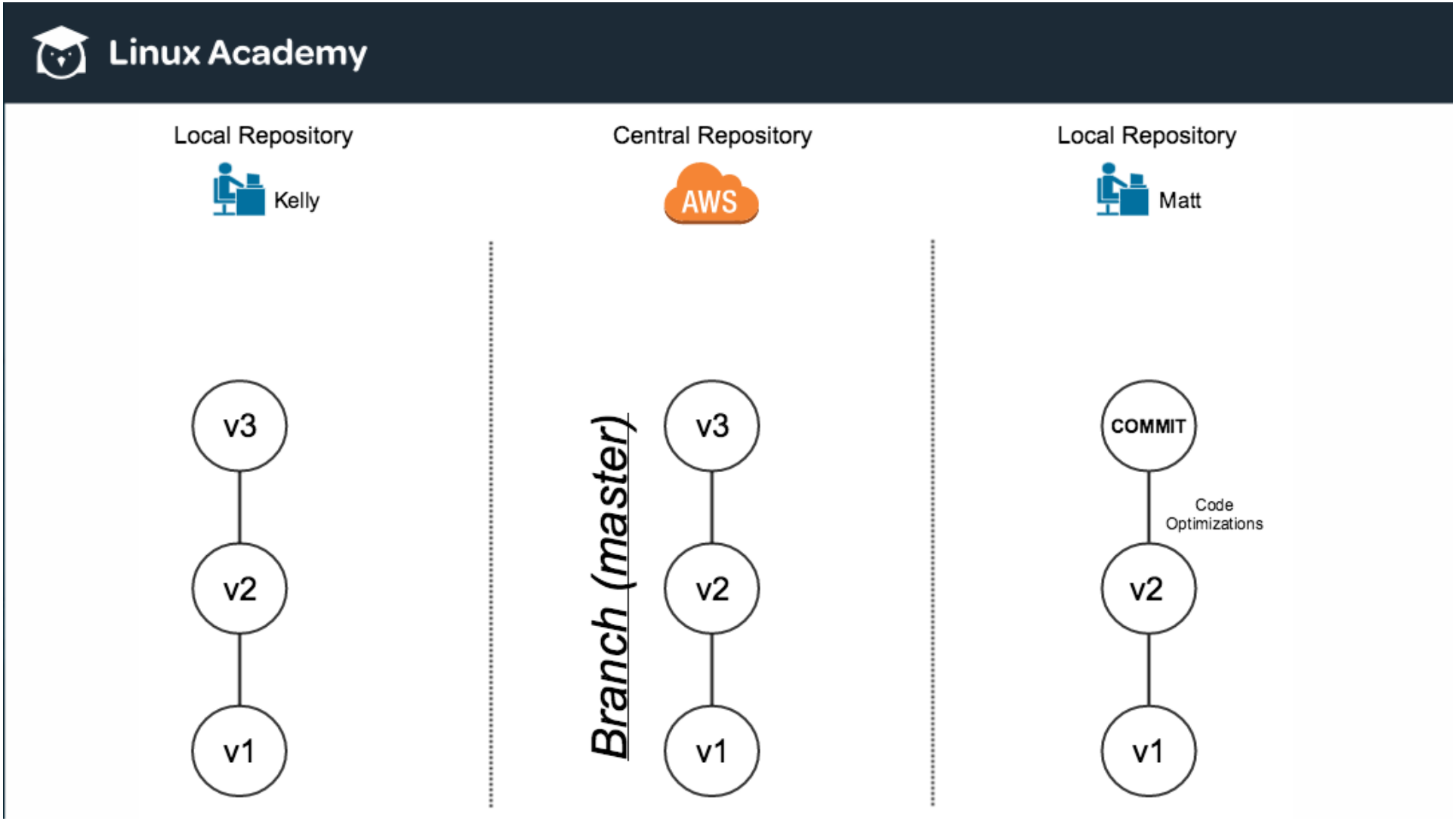
Linux Academy

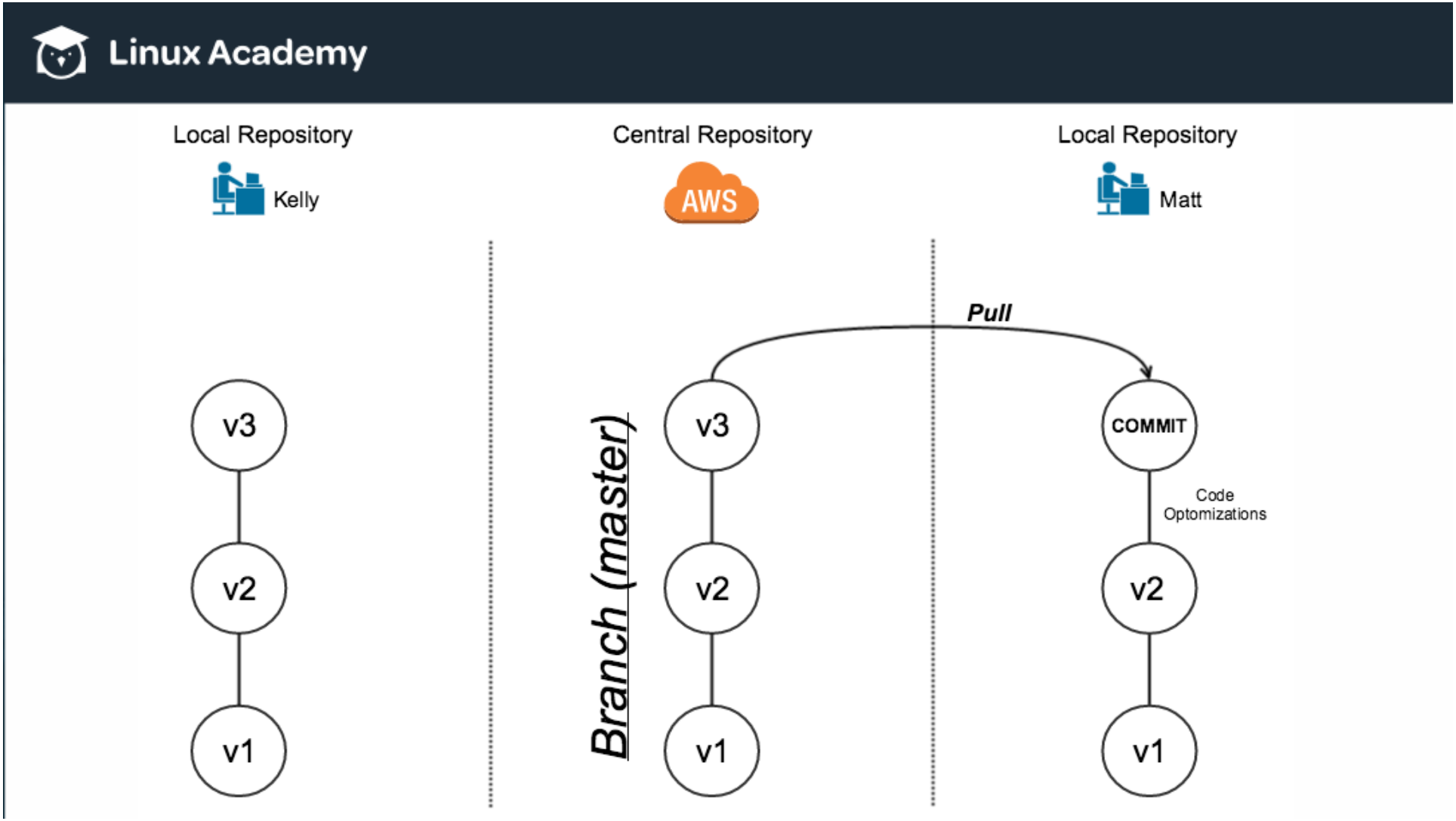


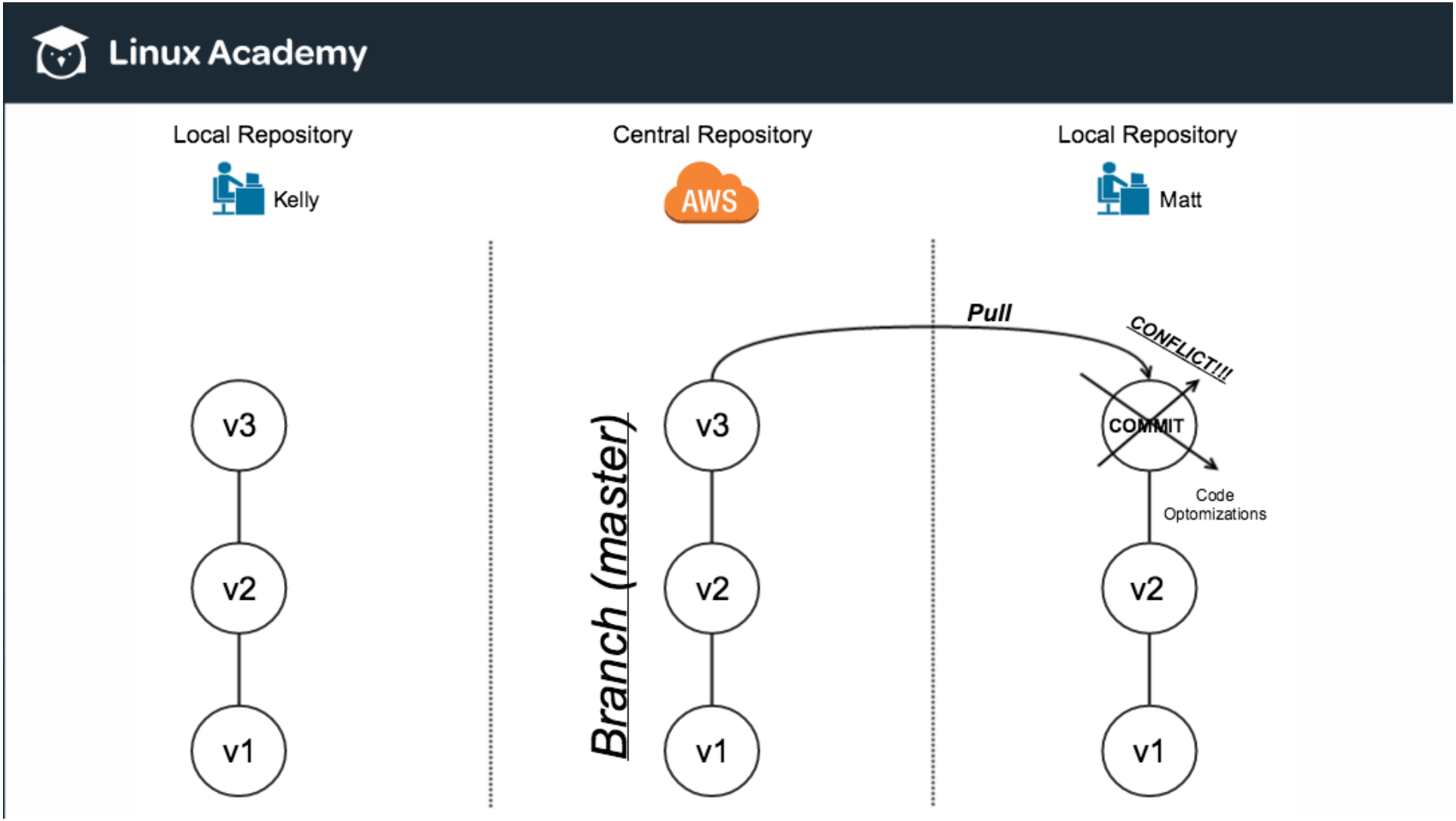

Linux Academy
Local Repository
Kelly
Central Repository
AWS
Local Repository
Matt
v3
v2
v1
Branch (master)
v3
v2
v1
v2
v1


Linux Academy
Local Repository
Kelly
v3
v2
v1
Central Repository
AWS
v4a
v4b
v3
v2
v1
Branch (master)
Local Repository
Matt
v2
v1


Linux Academy
Local Repository
Kelly
Central Repository
AWS
Local Repository
Matt
v3
v2
v1
Branch (master)
v3
v2
v1
v2
v1


Linux Academy
Local Repository
Kelly
Central Repository
AWS
Local Repository
Matt
v3
v2
v1
Branch (master)
v3
v2
v1
COMMIT
Code Optimizations
v2
v1


Linux Academy
Local Repository
Kelly
Central Repository
AWS
Local Repository
Matt
v3
v2
v1
Branch (master)
v3
v2
v1
Pull
COMMIT
Code
Optomizations
v2
v1


Linux Academy
Local Repository
Kelly
Central Repository
AWS
Local Repository
Matt
v3
v2
v1
Branch (master)
v3
v2
v1
Pull
CONFLICT!!!
COMMIT
Code
Optomizations
v2
v1

Setup & Configuration

- Tools we need to use CodeCommit
  - AWS CLI -> create, edit, delete, and view Repositories
  - Git -> clone, commit, push, pull, and create branches

- Communication Protocols
  - SSH
  - HTTPS

Pricing

- Free Teir (first 5 active users)
  - Unlimited repositories
  - 50 GB per month of storage
  - 10,000 Git request per month

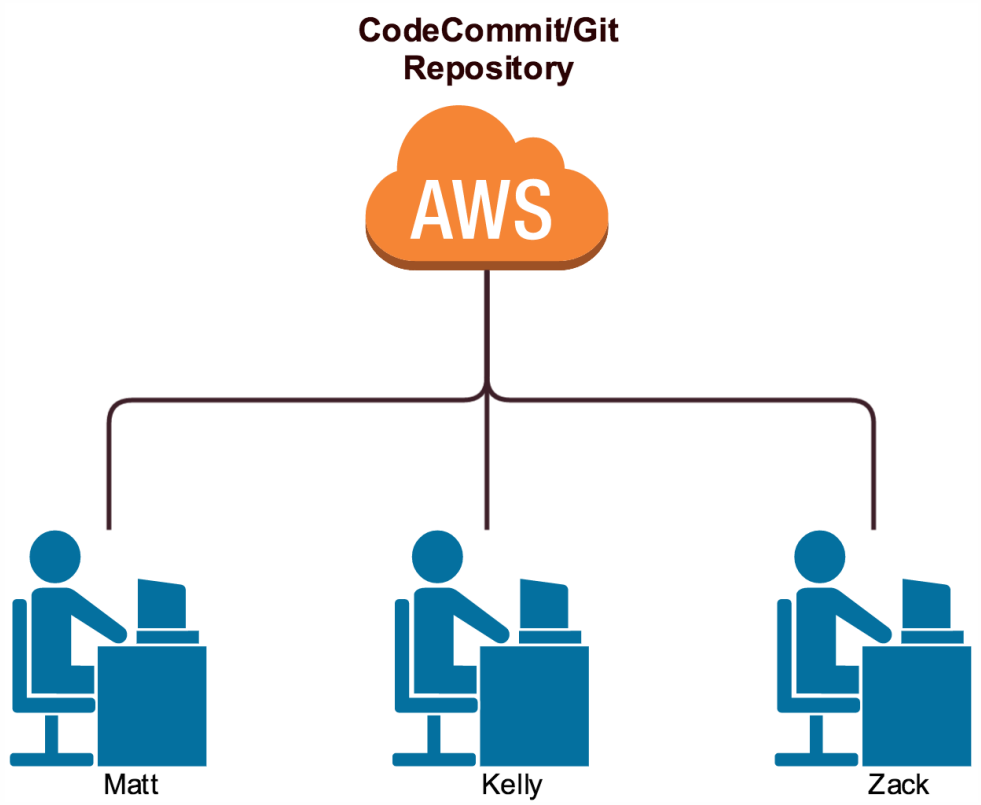
- $1 Per Month (each active user above the first 5)
  - Unlimited repositories
  - 10 GB per month of storage per active user
  - 2,000 Git request per month per active user

  For Example: If you have 8 active users, you totally bill will be $3, assuming you don't exceed any of the above limits.

- Overage
  - $0.06 per GB per month
  - $0.001 per Git request

Linux Academy

WonderWidgets Development Team




Linux Academy
Amazon Web Services
Developer Tools: CodeCommit Setup & Configuration

Choosing Your Setup & Configuration Videos

- Windows (HTTPS)
  - Video 1
  - Video 2
  - Video 3
- Windows (SSH)
  - Video 1
  - Video 2
  - Video 4
- Mac OSX (HTTPS)
  - Video 1
  - Video 5
  - Video 7
  - Video 9
- Mac OSX (SSH)
  - Video 1
  - Video 5
  - Video 7
  - Video 8
- Linux (HTTPS)
  - Video 1
  - Video 6
  - Video 7
  - Video 9
- Linux (SSH)
  - Video 1
  - Video 6
  - Video 7
  - Video 8

HTTPS or SSH

- Functionally, both protocols are basically the same
  - git clone ***https***://git-codecommit.us-east-1.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo
  - git clone ***ssh***://git-codecommit.us-east-1.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo
- Storing and verifying credentials is a main point of difference
  - HTTPS: Credential Helper
  - SSH: RSA Key Pairs
- Port access (your firewall and network security)
  - HTTPS: Port 443
  - SSH: Port 22

HTTPS or SSH

- HTTPS Pros:
  - Simple credential management
  - All data transfers are encrypted
  - Firewalls are often setup to allow traffic through port 443
- HTTPS Cons:
  - MAC OSX Keychain issues
- SSH Pros:
  - SSH is efficient
  - All data transfers are encrypted
- SSH Cons:
  - Credential management can be slightly more cumbersome
  - Firewalls can sometimes block port 22

HTTPS or SSH: Recommendation

- Windows & Linux: HTTPS
- Mac OSX: SSH (due to keychain issues)

Choosing Your Setup & Configuration Videos

- Windows (HTTPS)
  - Video 1
  - Video 2
  - Video 3
- Windows (SSH)
  - Video 1
  - Video 2
  - Video 4
- Mac OSX (HTTPS)
  - Video 1
  - Video 5
  - Video 7
  - Video 9
- Mac OSX (SSH)
  - Video 1
  - Video 5
  - Video 7
  - Video 8
- Linux (HTTPS)
  - Video 1
  - Video 6
  - Video 7
  - Video 9
- Linux (SSH)
  - Video 1
  - Video 6
  - Video 7
  - Video 8

Amazon Web Services

CodeCommit Basics:
Cloning, Commit, Push & Pull

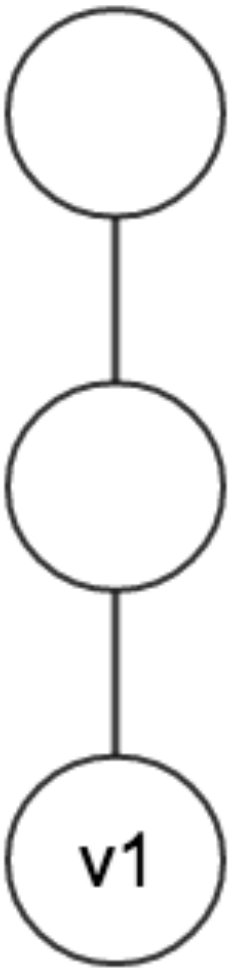
Cloning a Repository

Central Repository

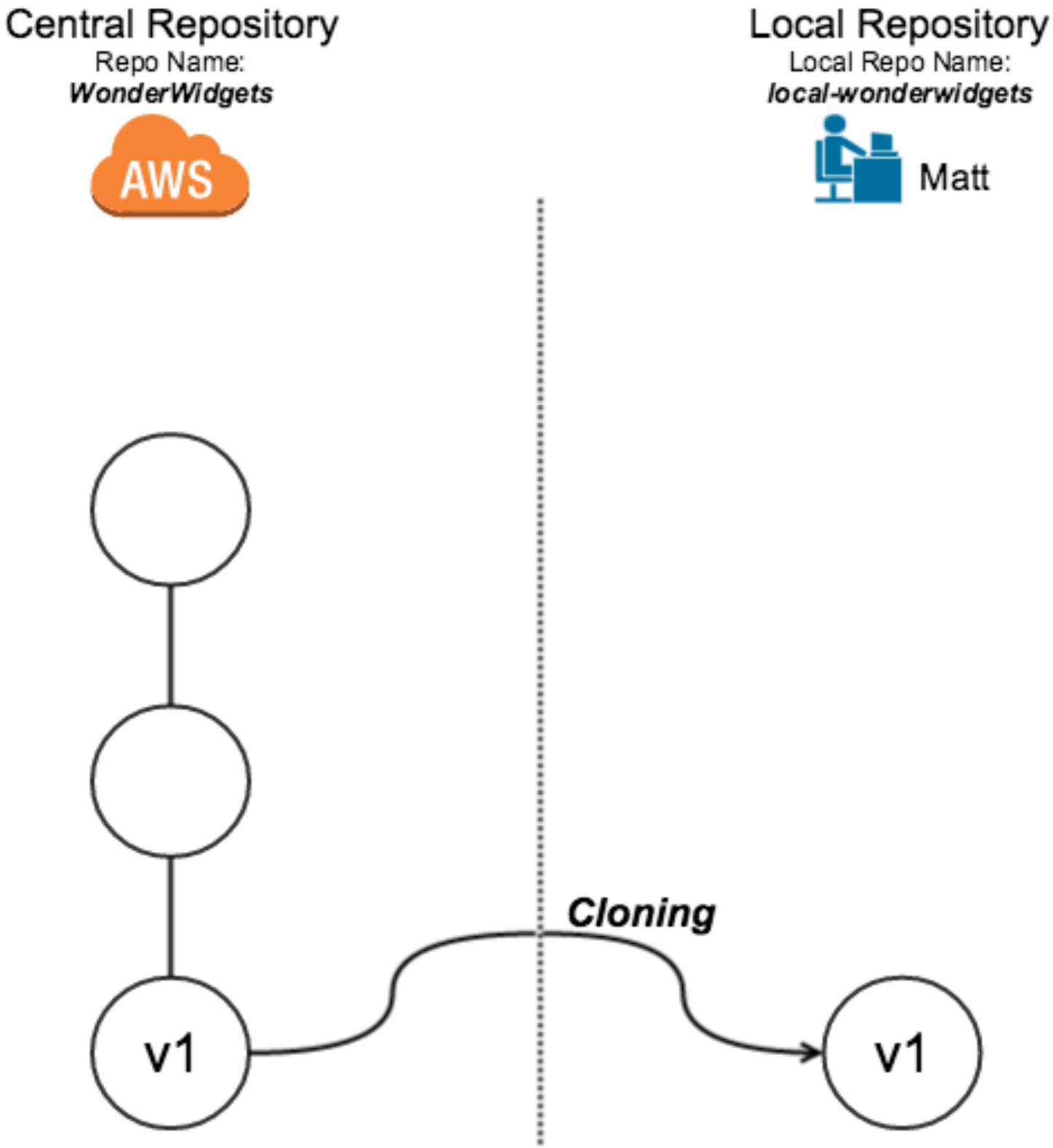
Repo Name:
WonderWidgets

Local Repository

Matt



Cloning a Repository



Cloning a Repository

- Definition:
  Cloning a repository creates an exact replica of the selected repository on your local machine.

- Related Base Command:
  git clone

- Base Required Attributes & Syntax:
  git clone <repository URL> <local repository name>

- Repository URL:
  Is provided by AWS, and is found in the AWS console by clicking on the repository OR by running the AWS CLI command
  aws codecommit get-repository --repository-name <Name>

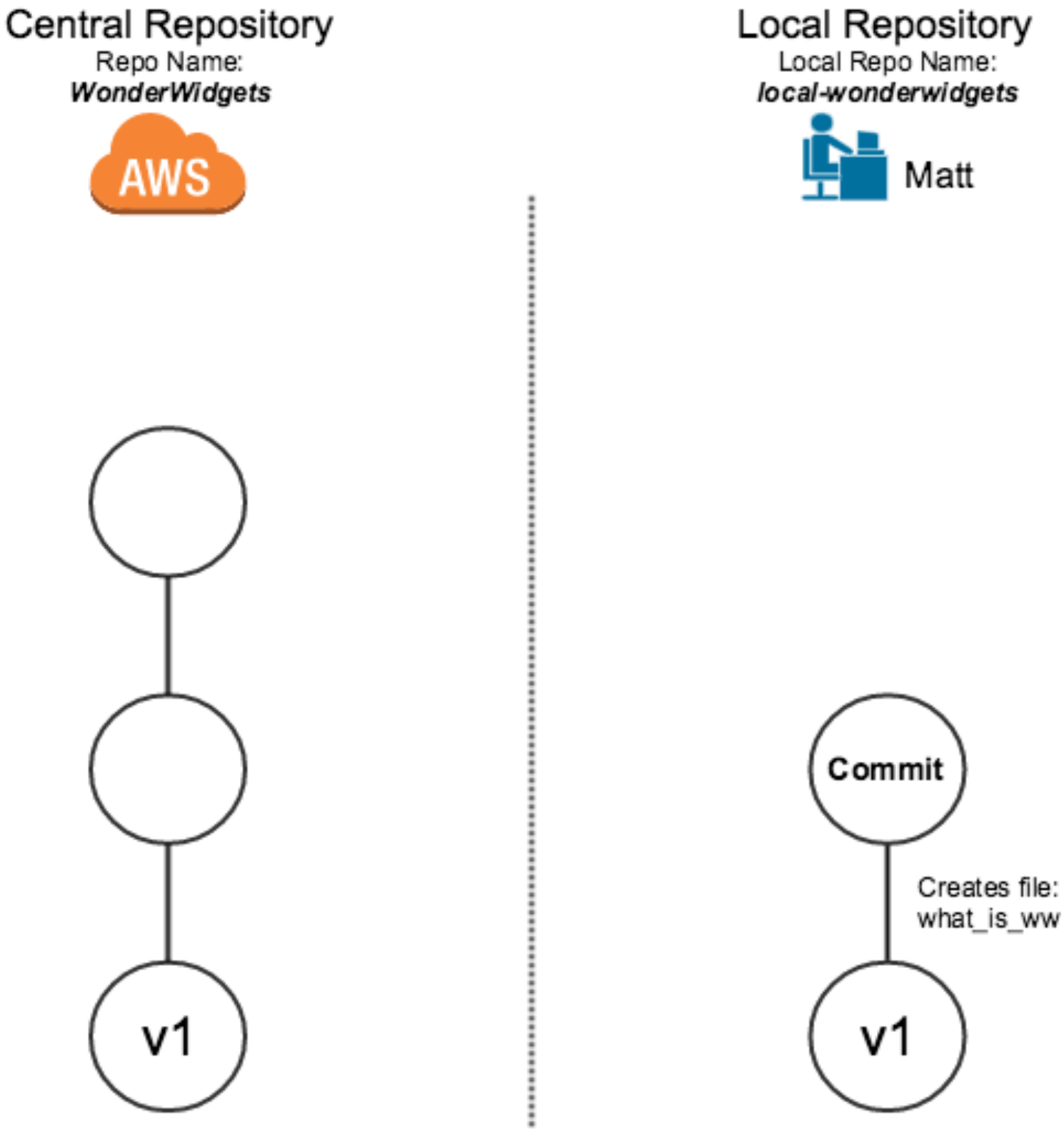
- Local Repository Name:
  Whatever name you want your local repository directory to have. The cloning command will create this directory for you.

- Example:

git clone https://git-codecommit.us-east-1.amazonaws.com/v1/repos/WonderWidgets local-wonderwidgets

- For Windows SSH users:
  git clone ssh://Your-SSH-Key-ID@git-codecommit.us……

Commits


Central Repository
Repo Name:
WonderWidgets
AWS
Local Repository
Local Repo Name:
local-wonderwidgets
Matt
v1
Commit
Creates file:
what_is_ww
v1

Using Commits

- Definition:
  A Commit stores changes you have made to the repository, and allows you to sync those changes at a time of your choosing.

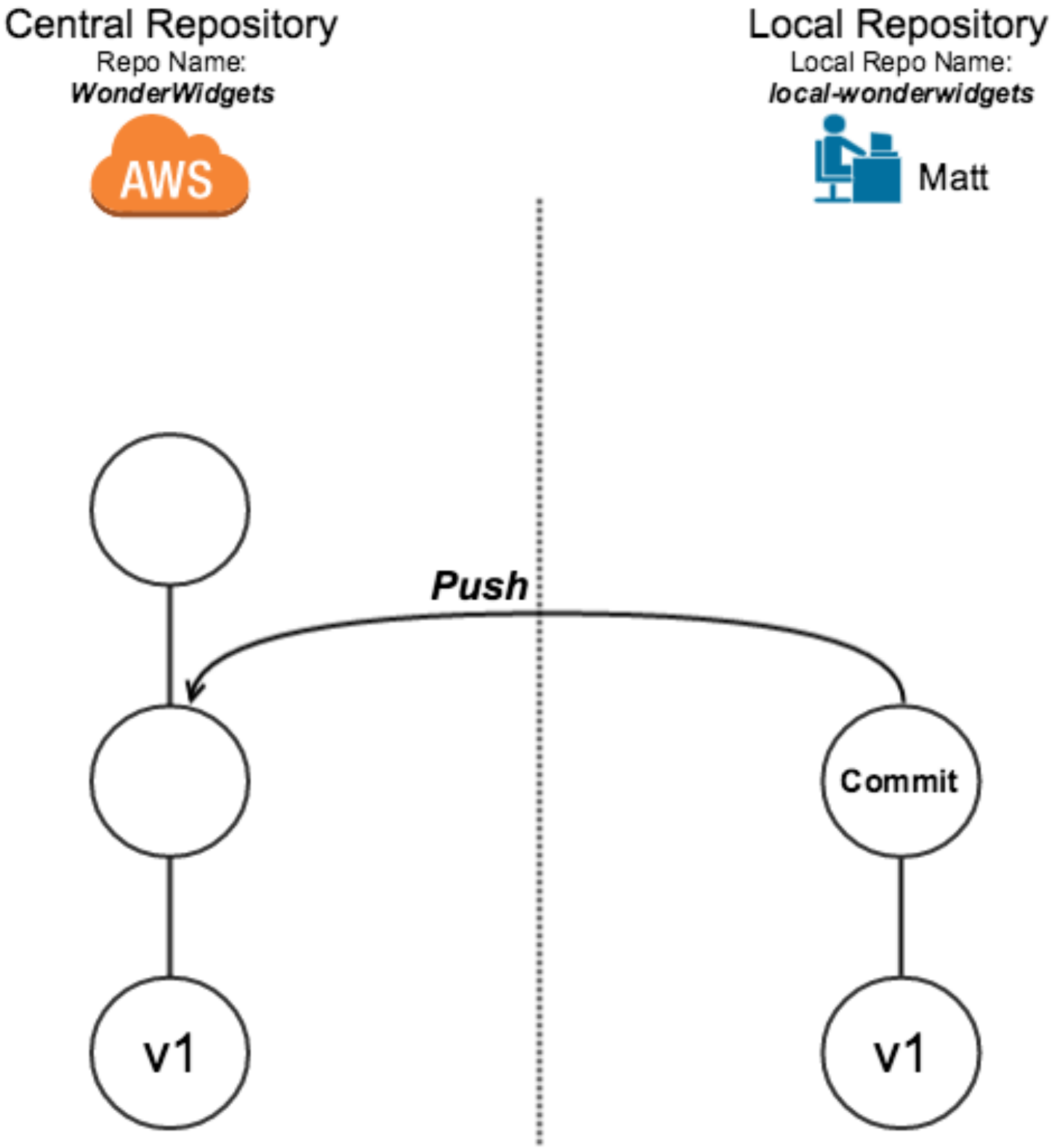
- Related Base Commands:
  git add *(adds a file to a commit)*
  git rm *(removes a files from a commit)*
  git status *(view files that have been added to the commit)*
  git commit *(finalize the commit)*

- Base Required Attributes & Syntax:
  git add <file name>
  git rm --cached <file name>

- Common Optional Flags:
  git commit -m *(Add a description to the commit)*
  git status -sb (View what has/hasn't been added to the pending commit in a concise format)

- Examples:
  git add what_is_ww
  git rm --cached what_is_ww
  git status
  git commit –m "added the what_is_ww file"

Pushing a Commit



Pushing a Commit

- Definition:
  Pushing a commit is how you sync your changes with the central repository.

- Related Base Commands:
  git remote (view remote name)
  git branch (view branch name)
  git diff --stat *(view which files will be pushed)*
  git push *(push the commit to the central repository)*

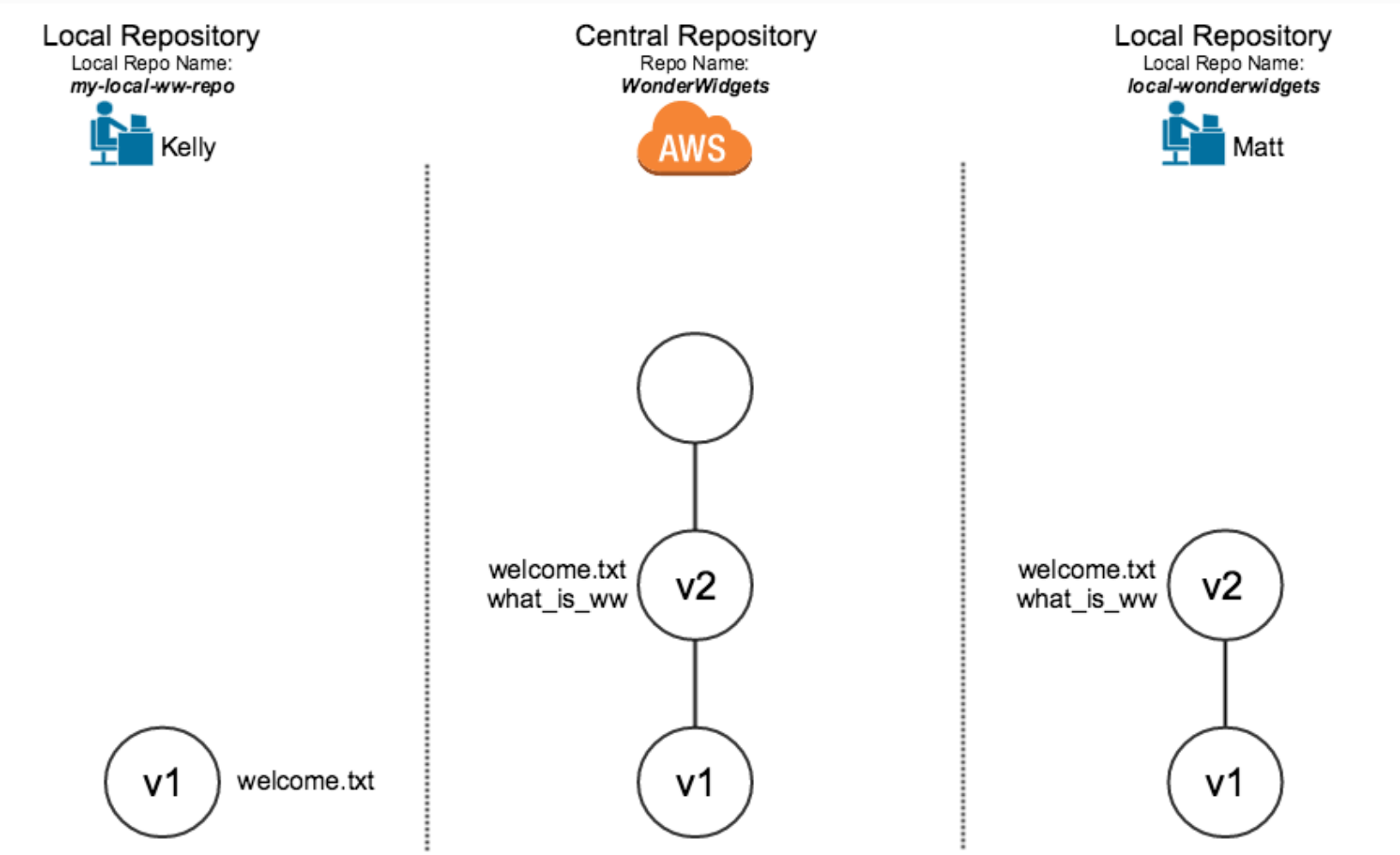
- Base Required Attributes & Syntax:
  git diff --stat <remote-name>/<branch-name>
  git push <remote-name> <branch-name>

- Remote Name:
  The "nickname" the local repo uses for the central repository.

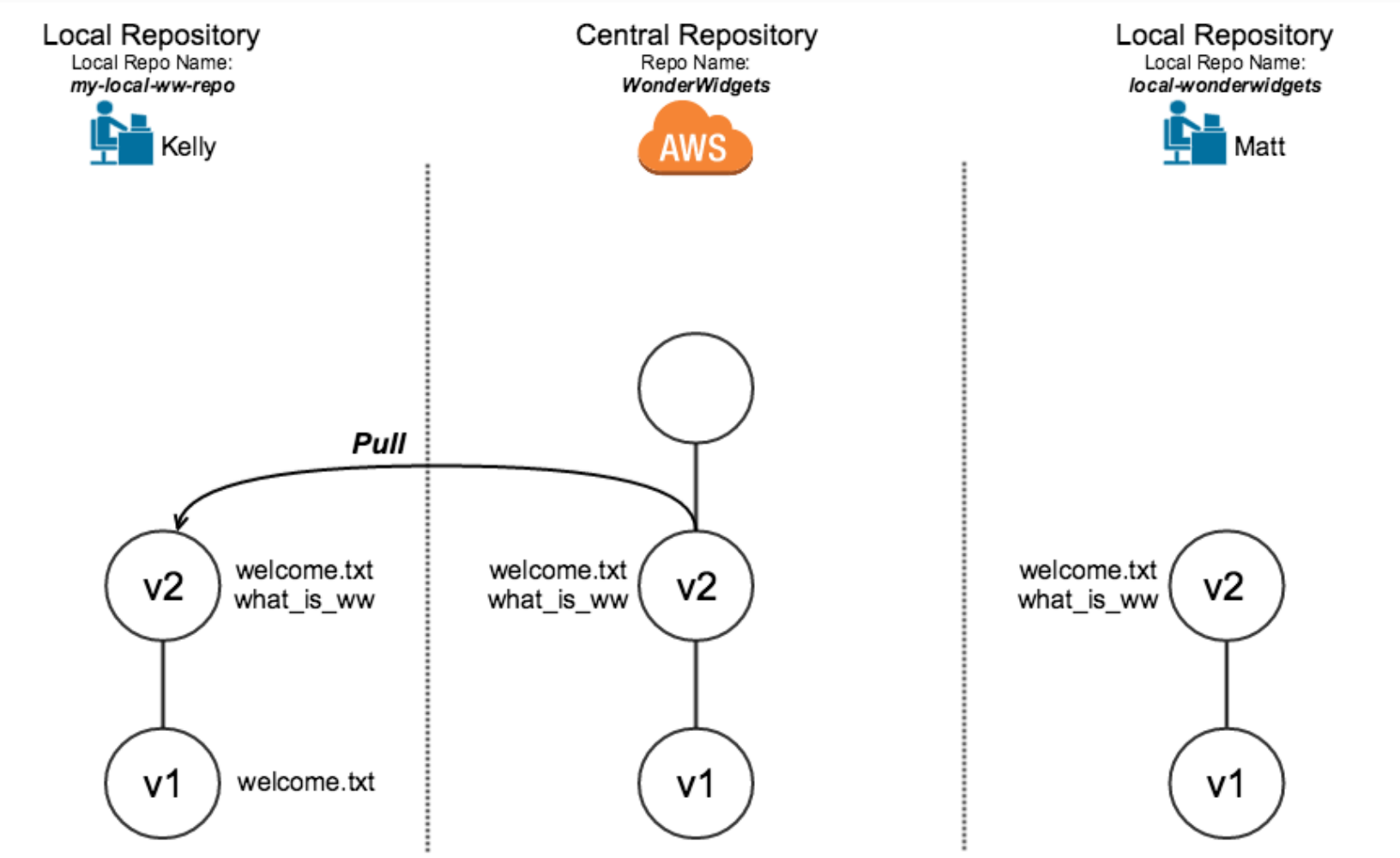
- Branch Name:
  The name of the branch you want to push the commit to.

- Examples:
  git remote
  git branch
  git diff --stat origin/master
  git push origin master

Pulling Updates



Pulling Updates



Pulling Updates

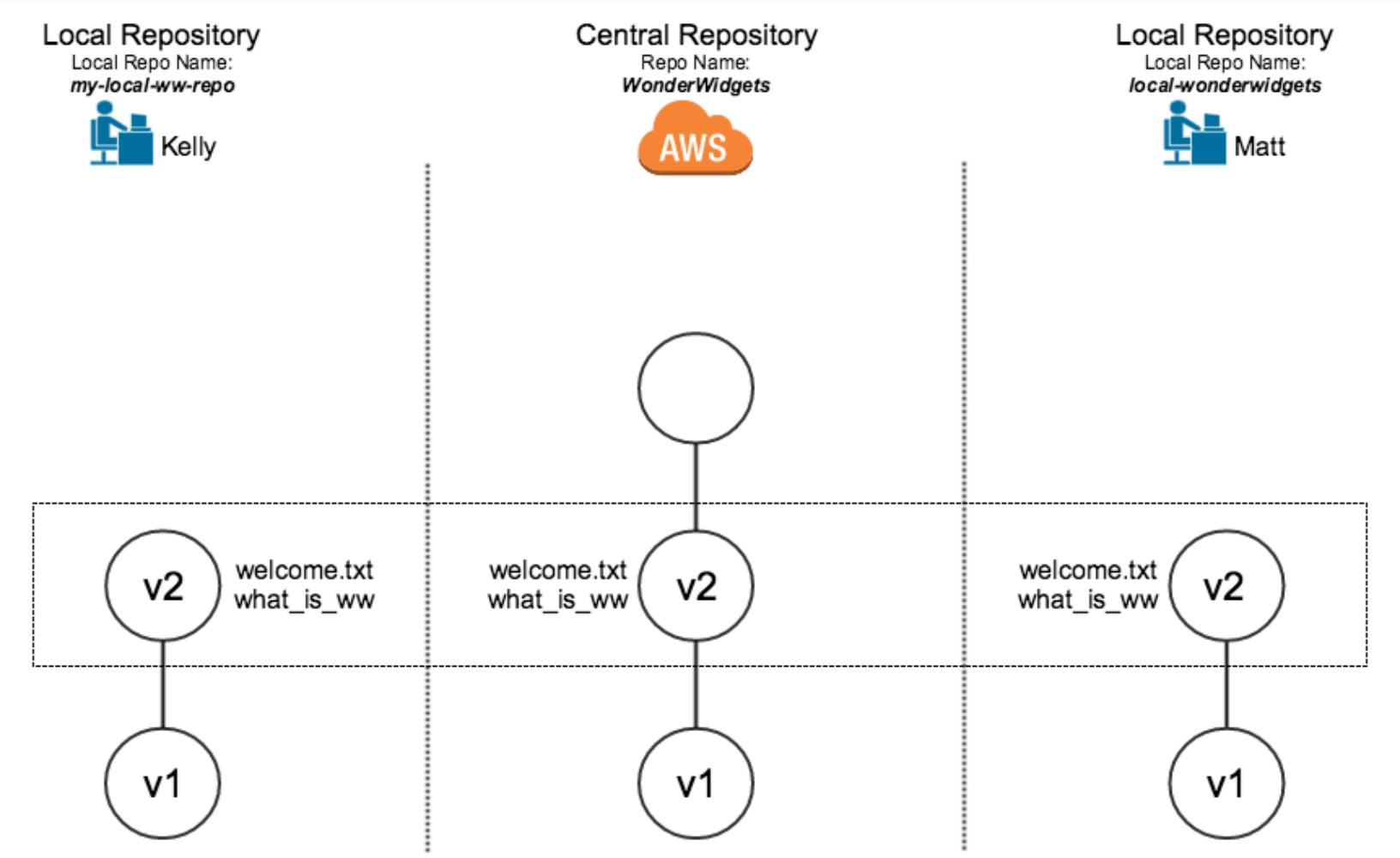
- Definition:
  Pulling downloads all the changes that have occurred in the central repository since your last sync (clone or pull).

- Base Commands:
  git pull

- Base Required Attributes & Syntax:
  git pull <remote-name> <branch-name>

- Example:
  git pull origin master

Everyone is up to date!



Linux Academy

Amazon Web Services

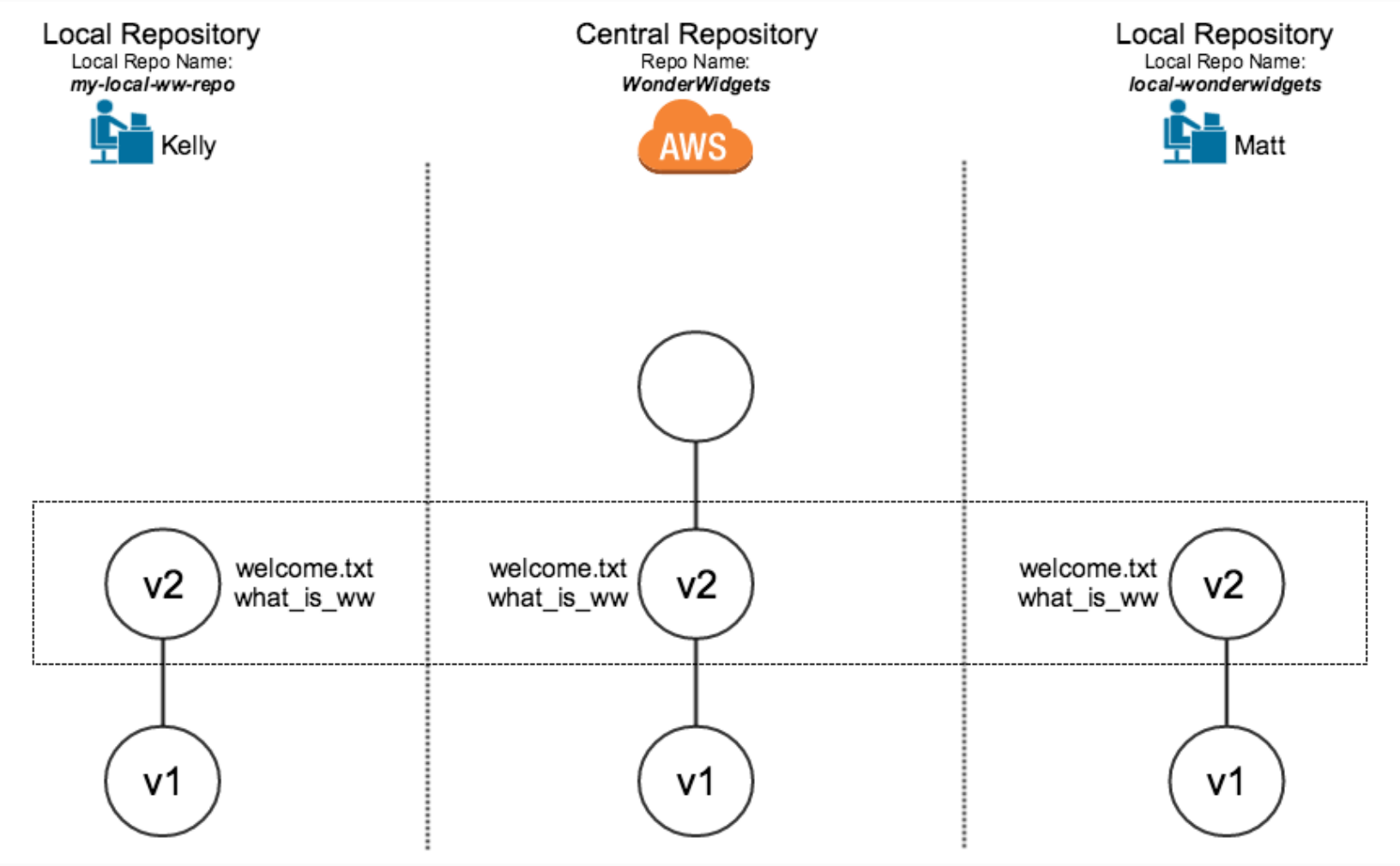
CodeCommit Basics:
Cloning, Commit, Push & Pull

Thank you for watching!

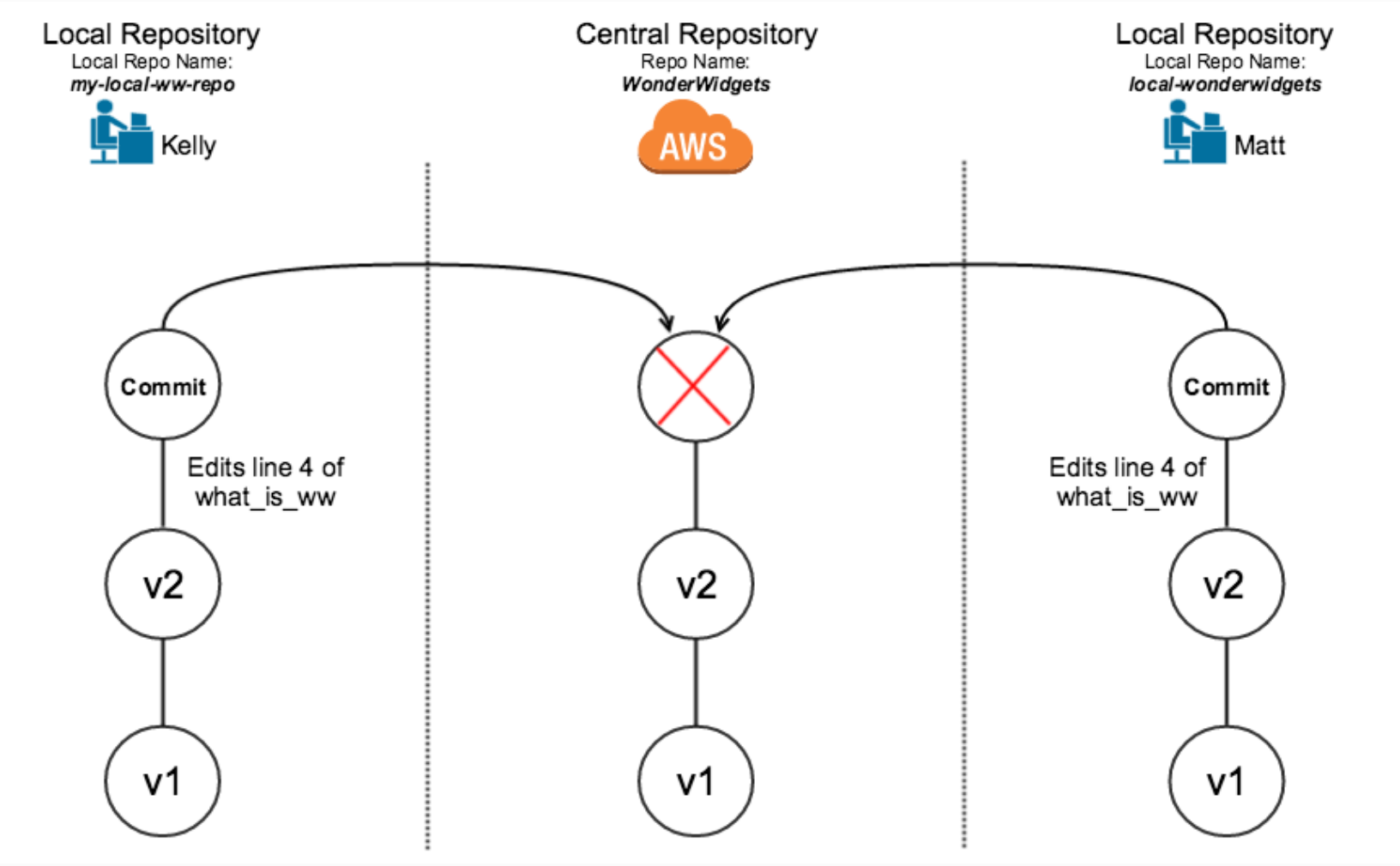
Amazon Web Services

CodeCommit Basics:
Merging Basic Conflicts

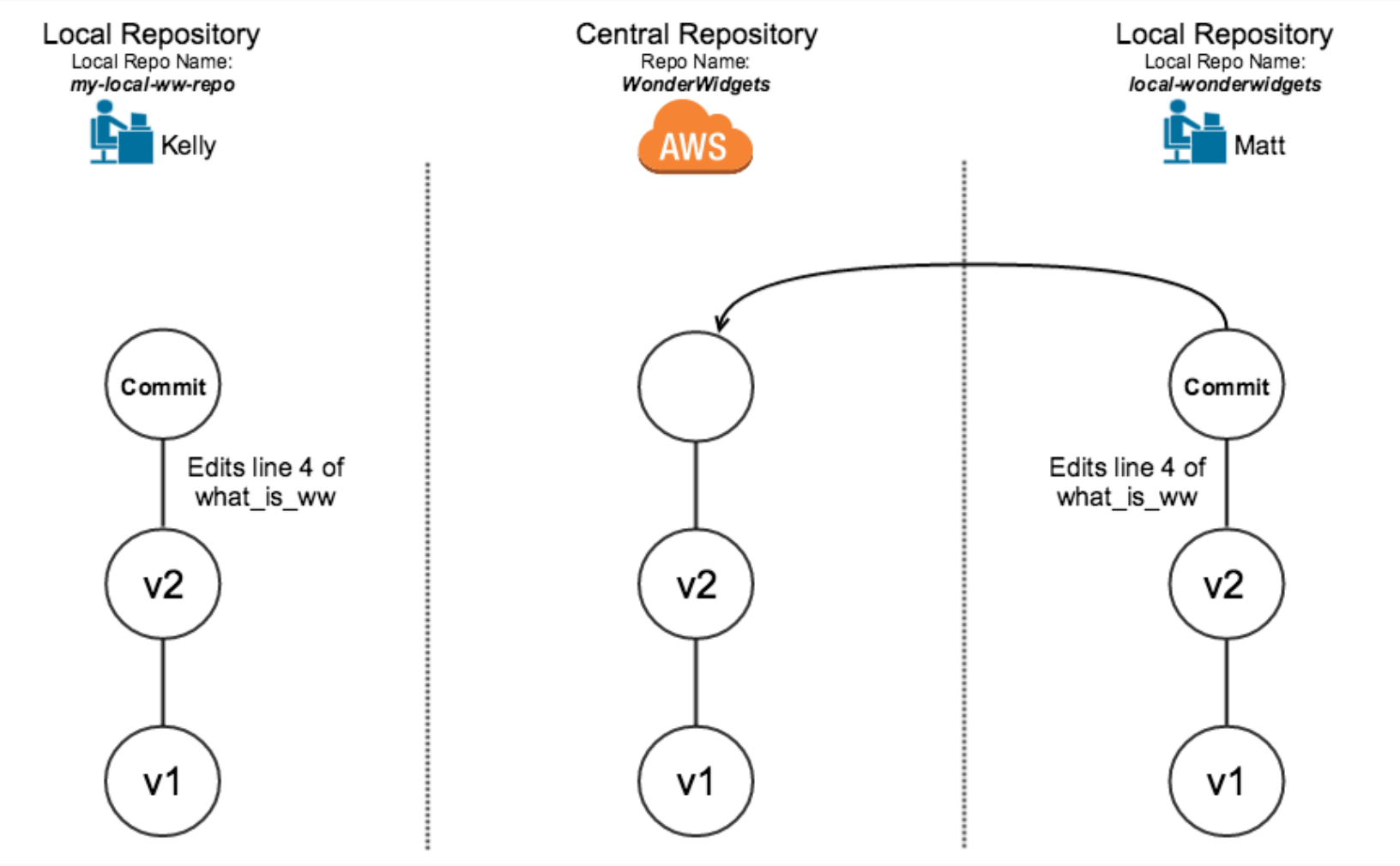
Current Repository Status



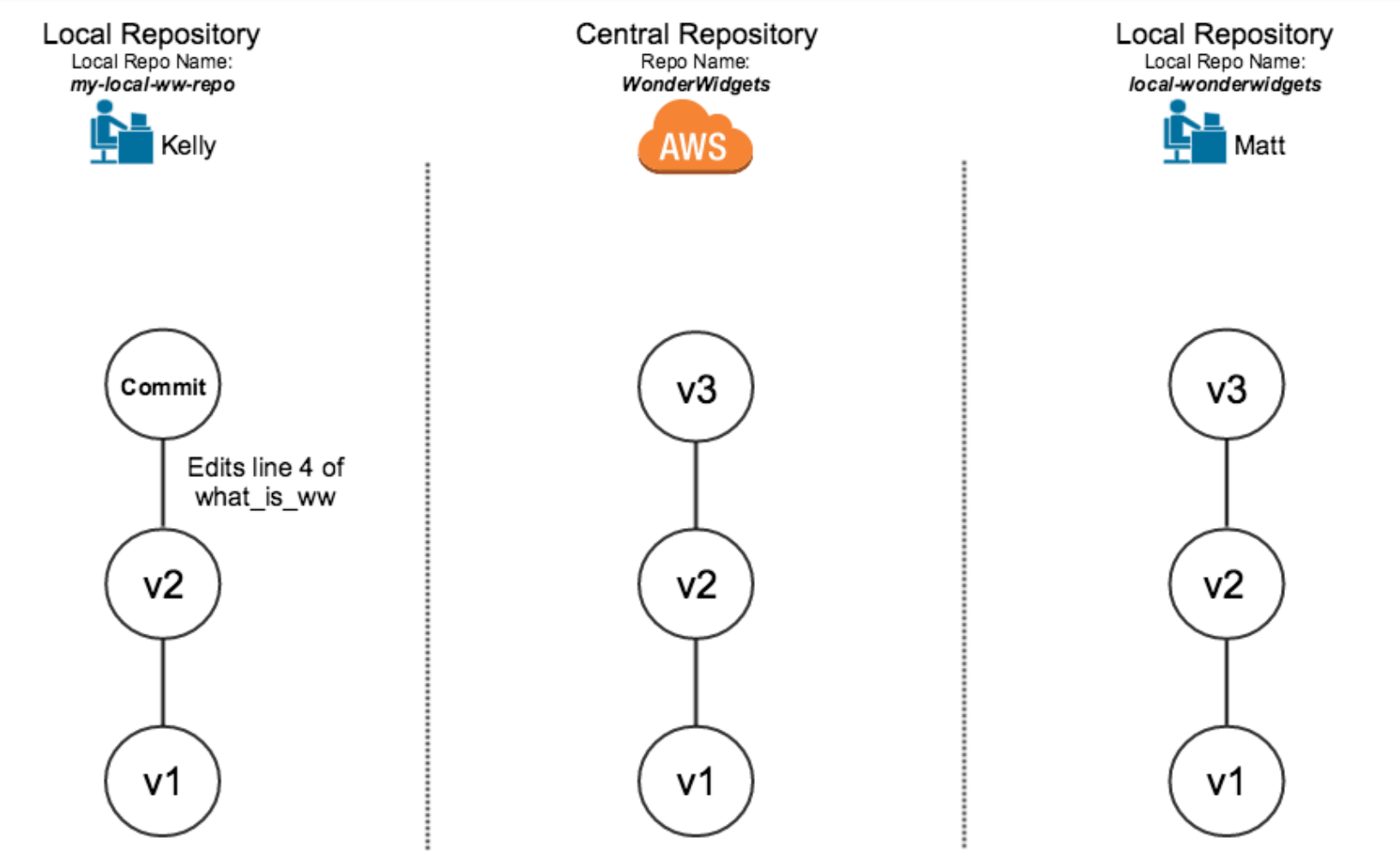
Understanding Basic Conflicts



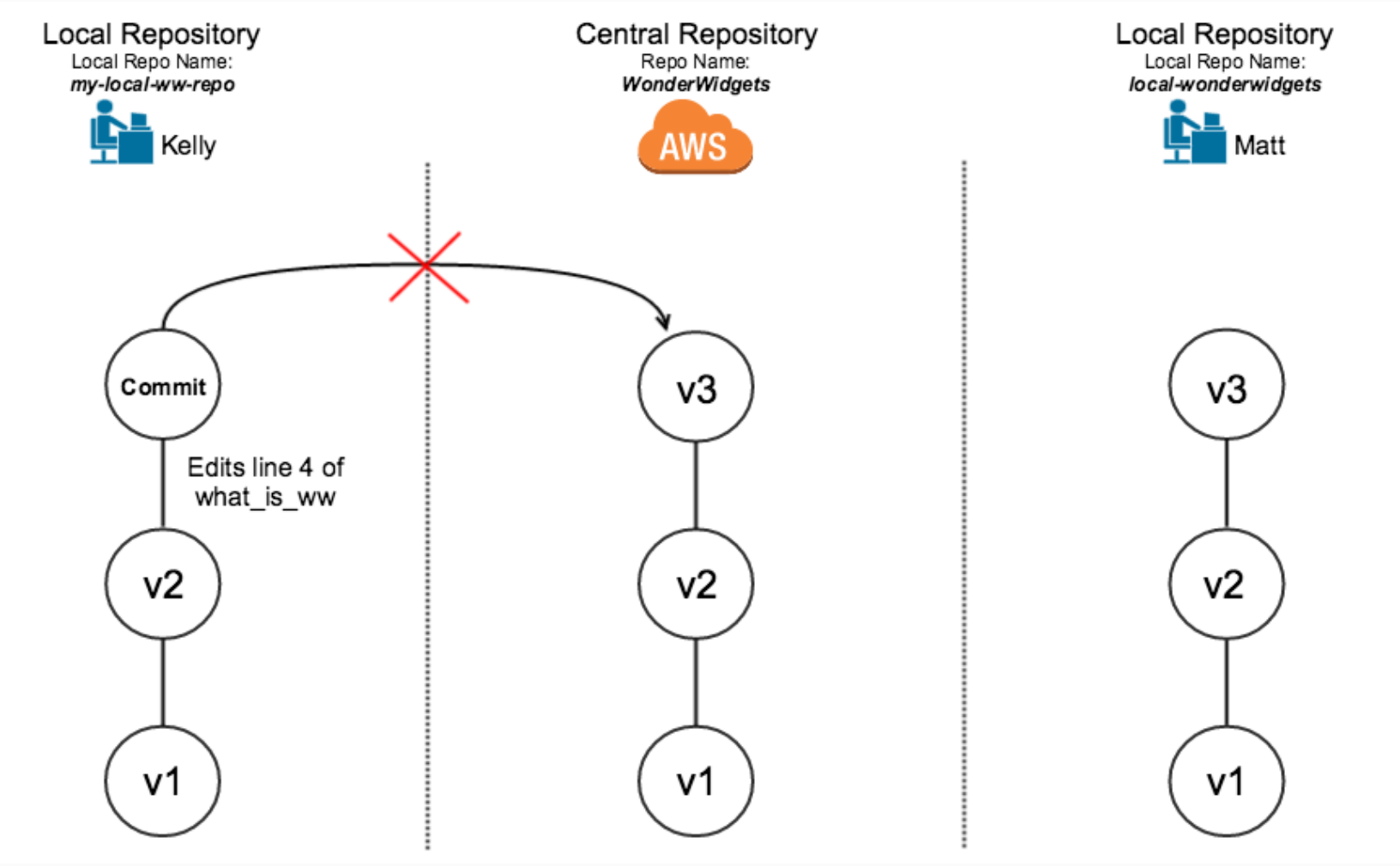
Understanding Basic Conflicts



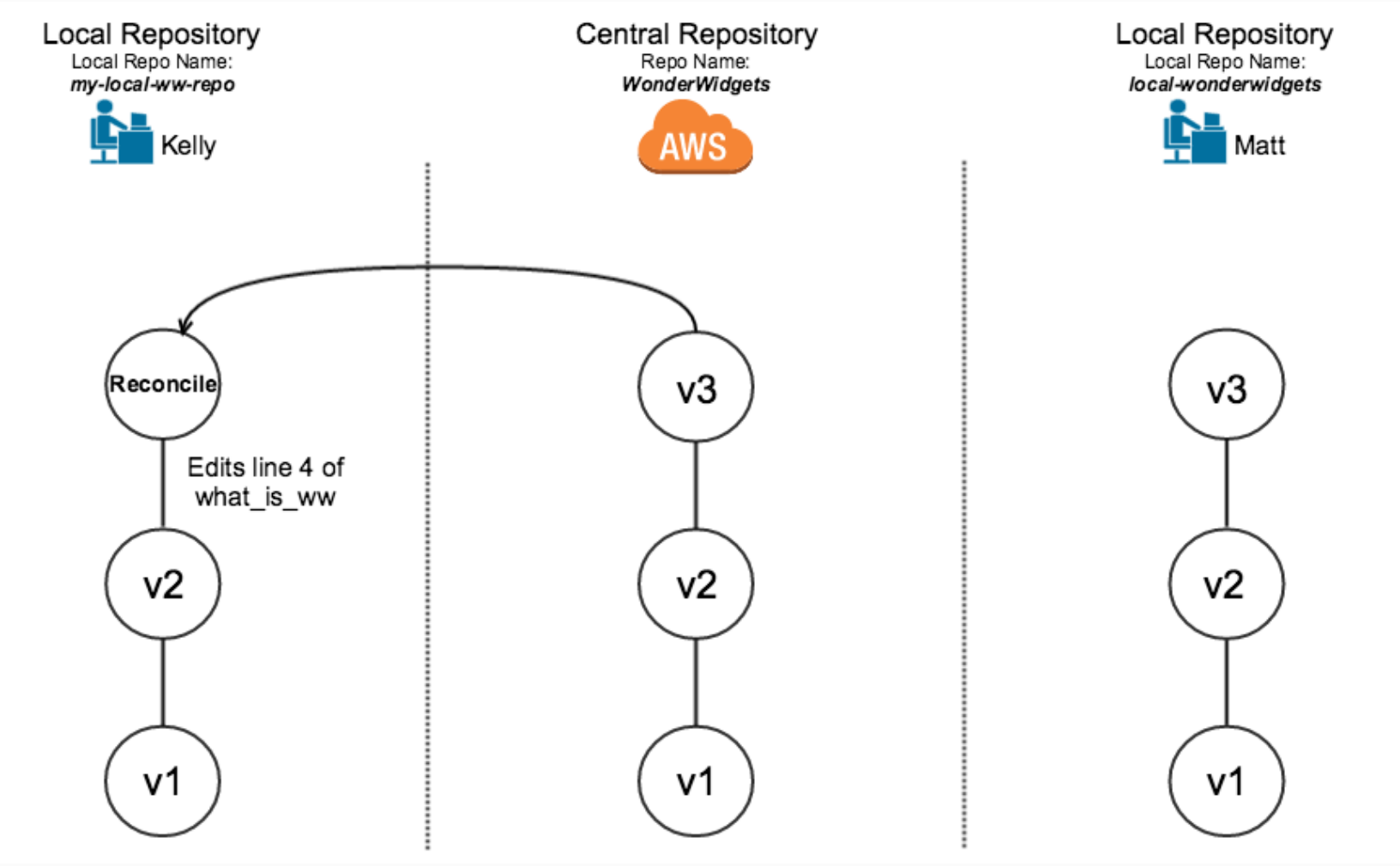
Understanding Basic Conflicts



Understanding Basic Conflicts



Understanding Basic Conflicts



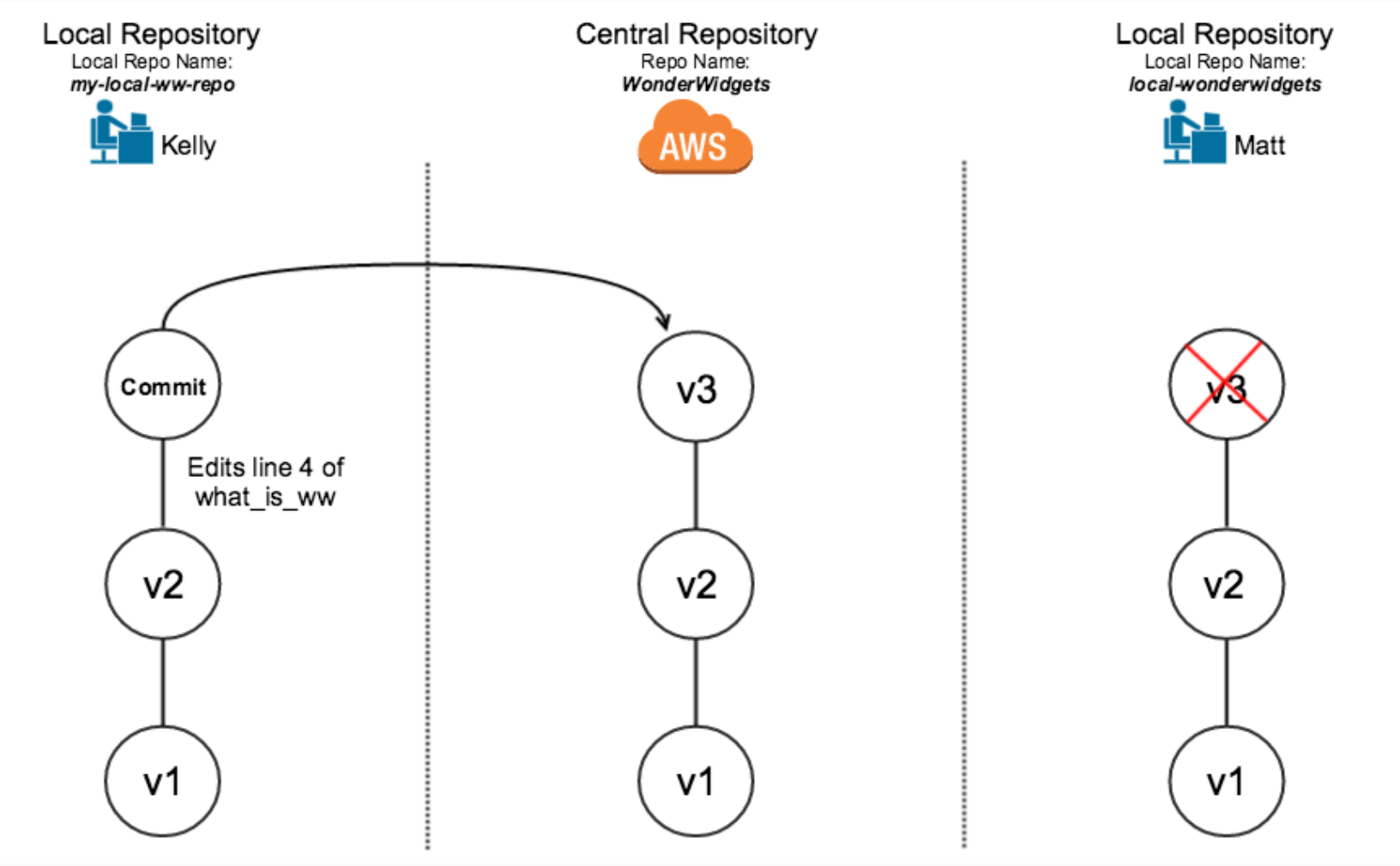
Understanding Basic Conflicts

- Definition:
  Conflicts occur when two or more users push a commit that has edits to the same line in a specific file.

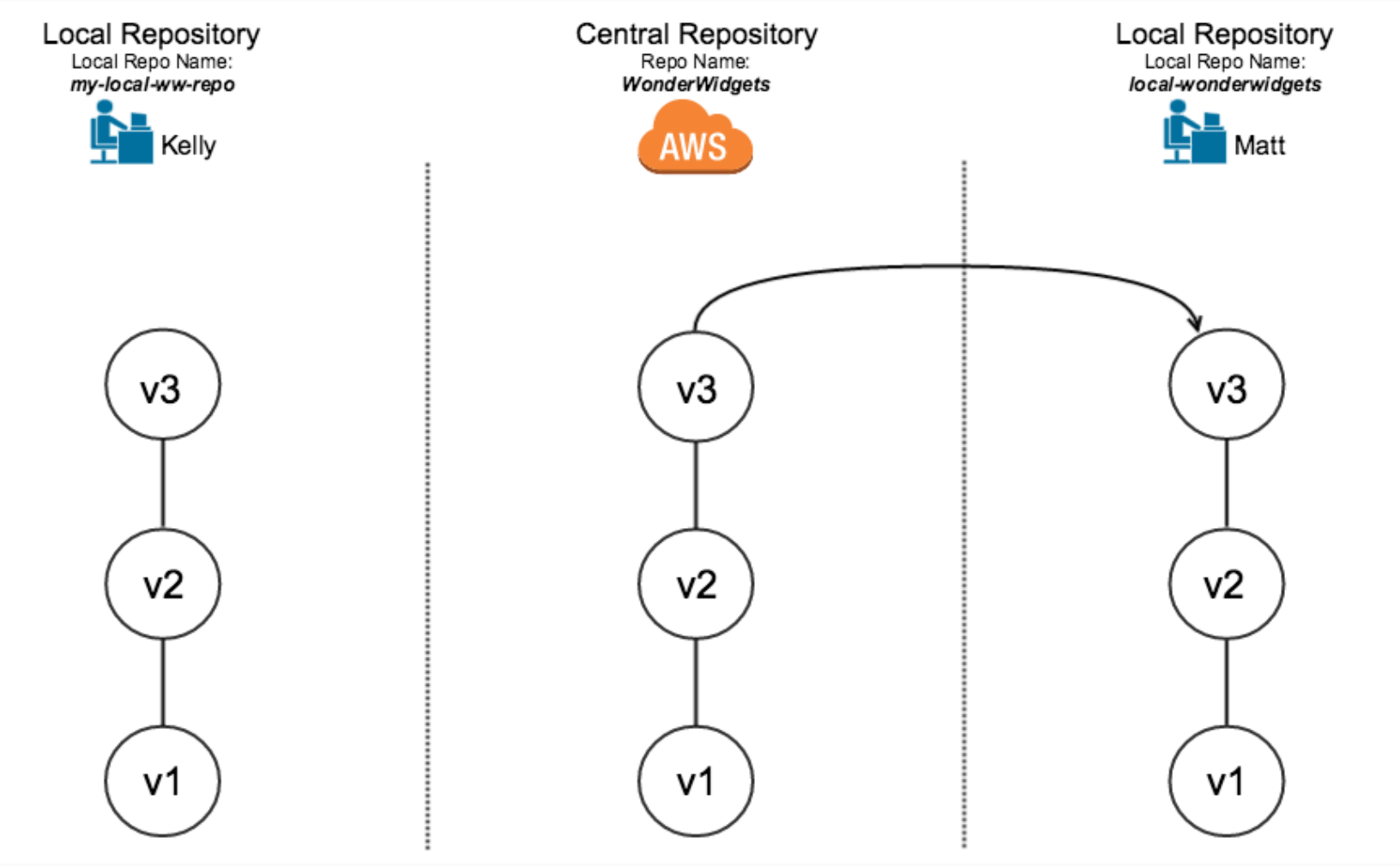
- HEAD Pointer:
  Refers to the most recent commit on the current branch. When you change branches, HEAD is updated to refer to the new branch's latest commit.

- Commit ID #:
  Hexadecimal number that identifies a commit
  Long form: ***f5c5cac0033439c17ebf905d4391dc0705dbd5f1***
  Short form: ***f5c5cac***

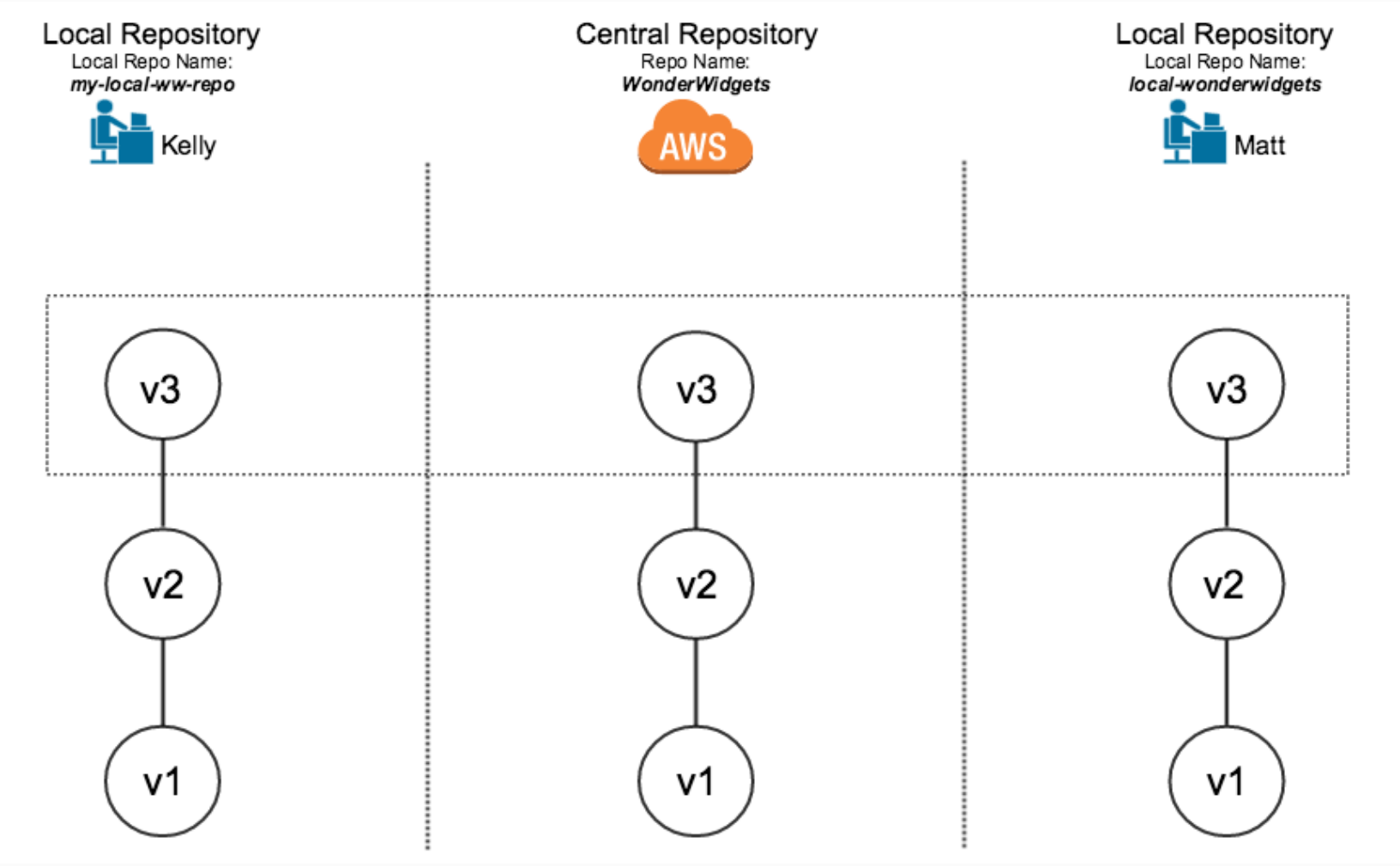
Understanding Basic Conflicts



Understanding Basic Conflicts



Understanding Basic Conflicts



Linux Academy

Amazon Web Services

CodeCommit Basics:
Merging Basic Conflicts

Thank you for watching!

Amazon Web Services

CodeCommit Basics:
Local Branches

Understanding Branches

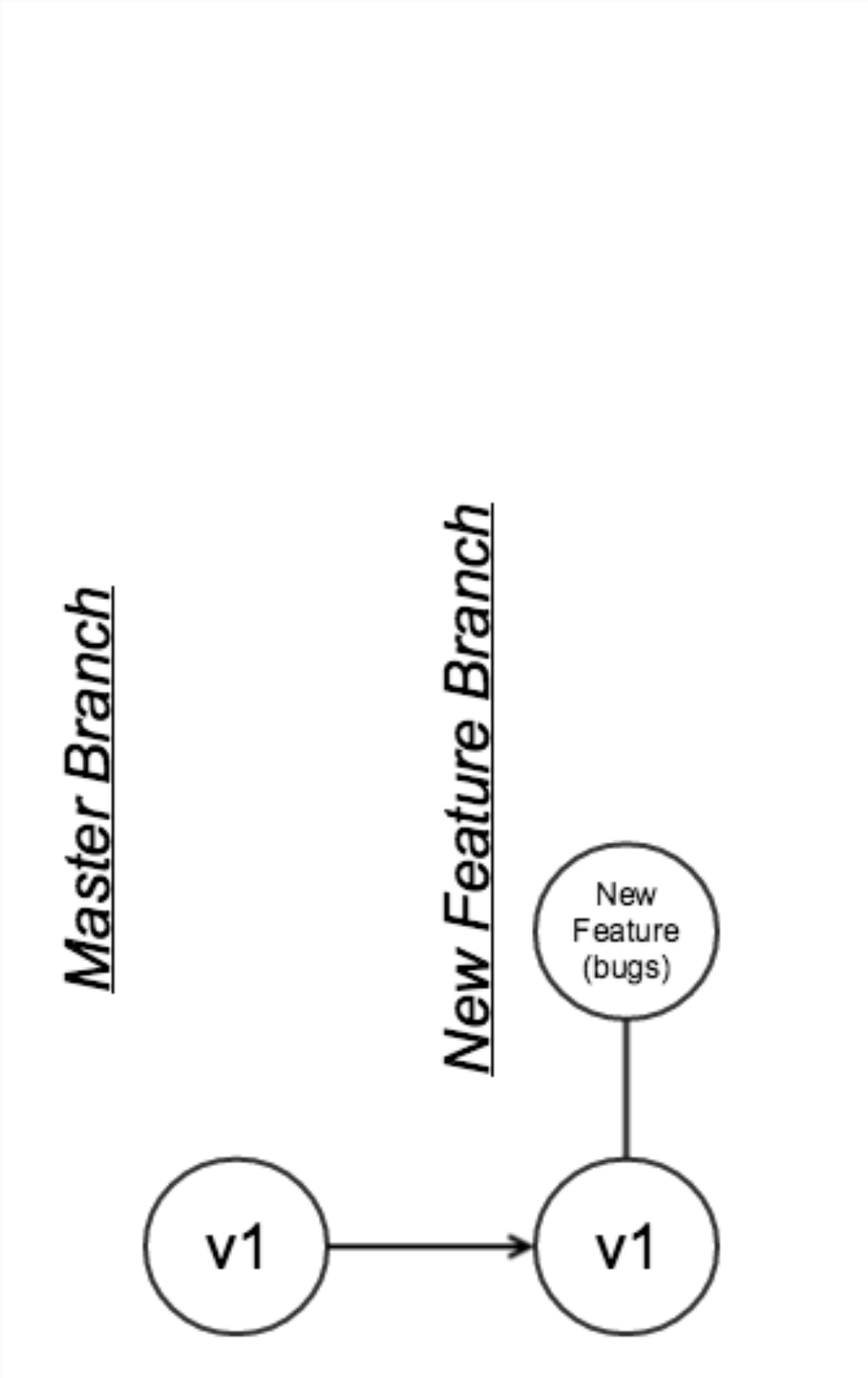
- Definition:
  A branch represents an independent line of development. Working on a branch (instead of on master), allows developers to safely make changes without disrupting the code on the master branch.

- Common examples of when branches are used:
  - *Building a new feature*
  - *Fixing a bug*

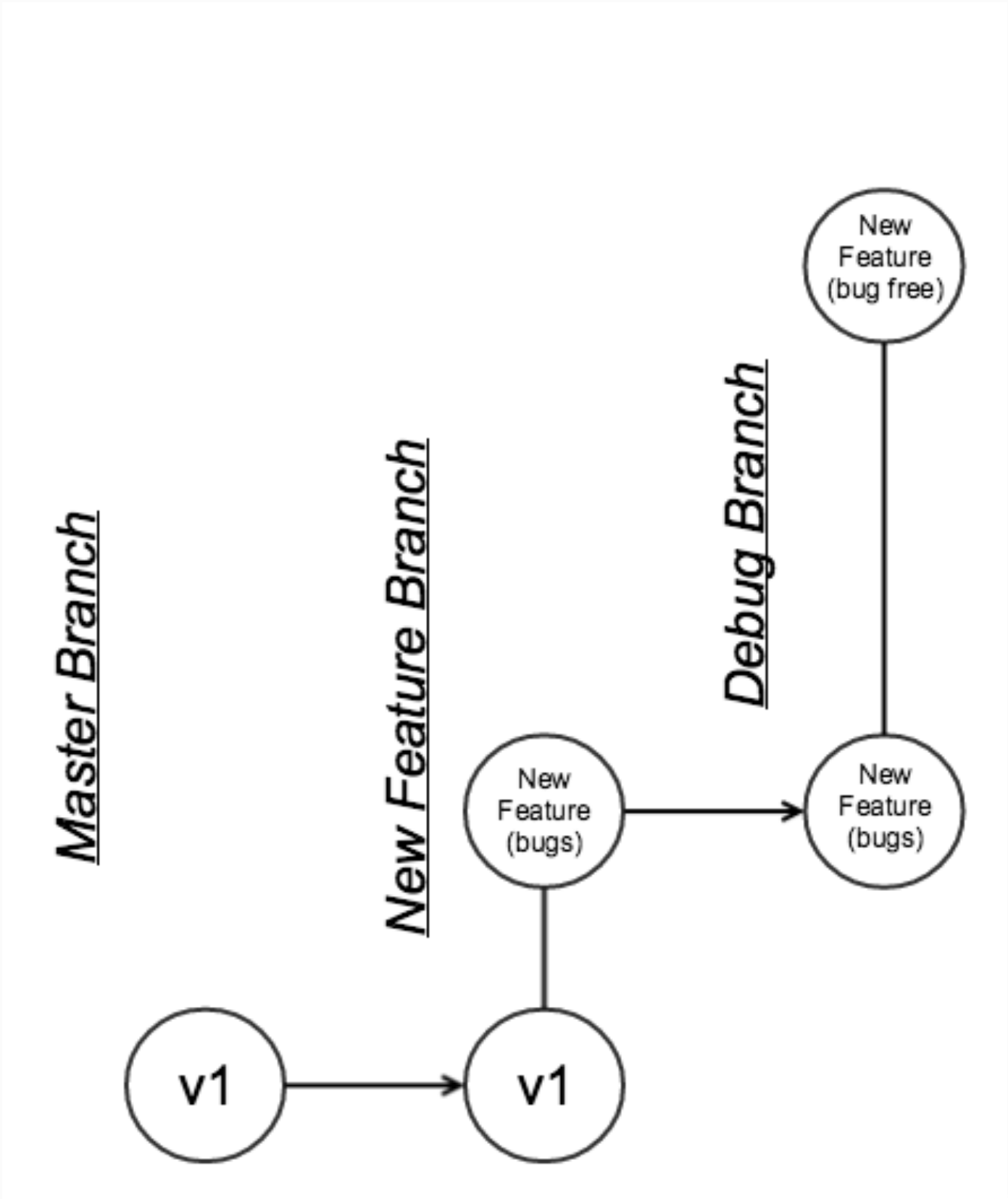
  ***Master**: Production Code*
  ***Branch**: Development Code*

- Managing Local Branches:
  - *Create*
  - *View*
  - *Switch*
  - *Merge*
  - *Delete*

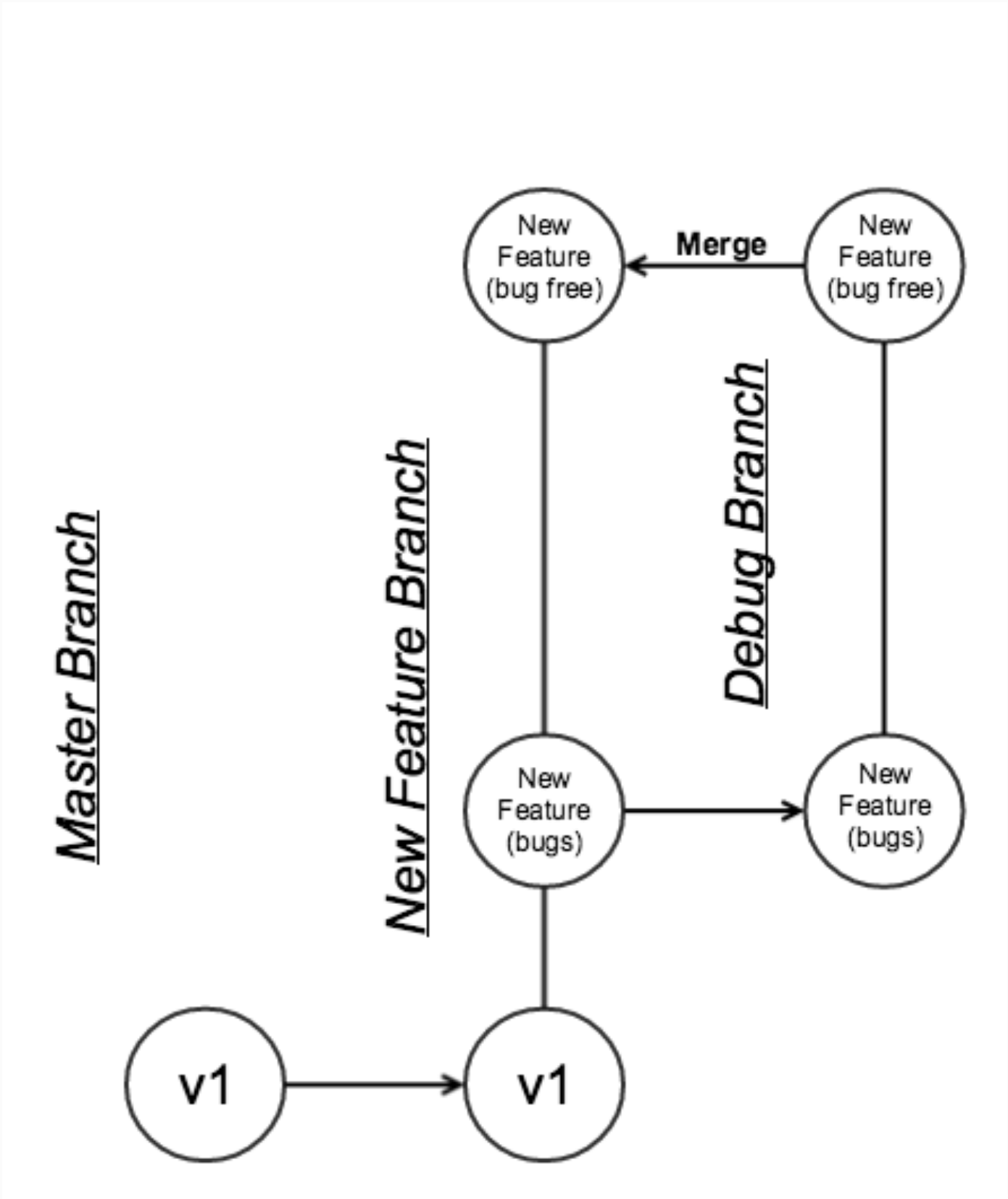
Understanding Branches



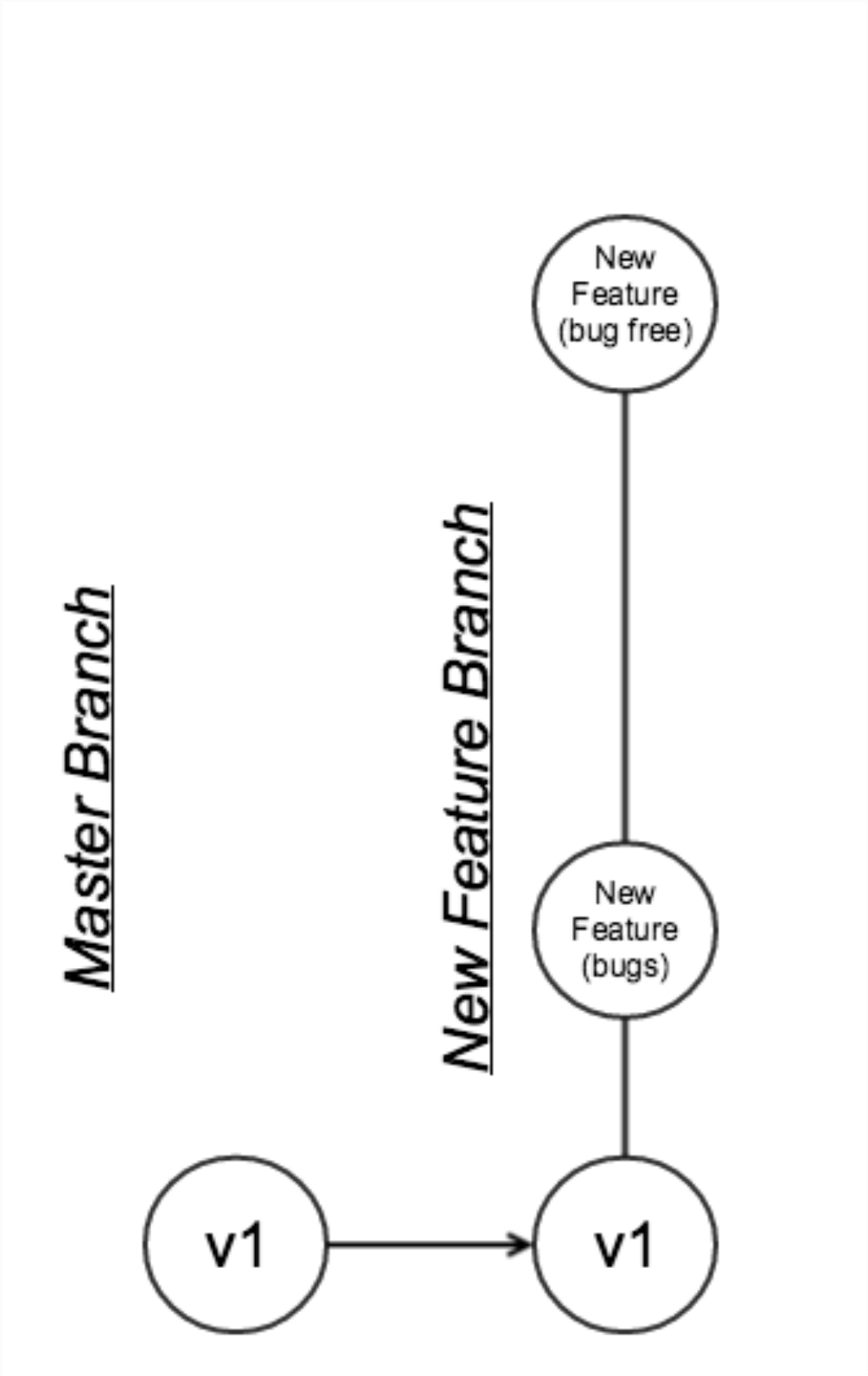
Understanding Branches



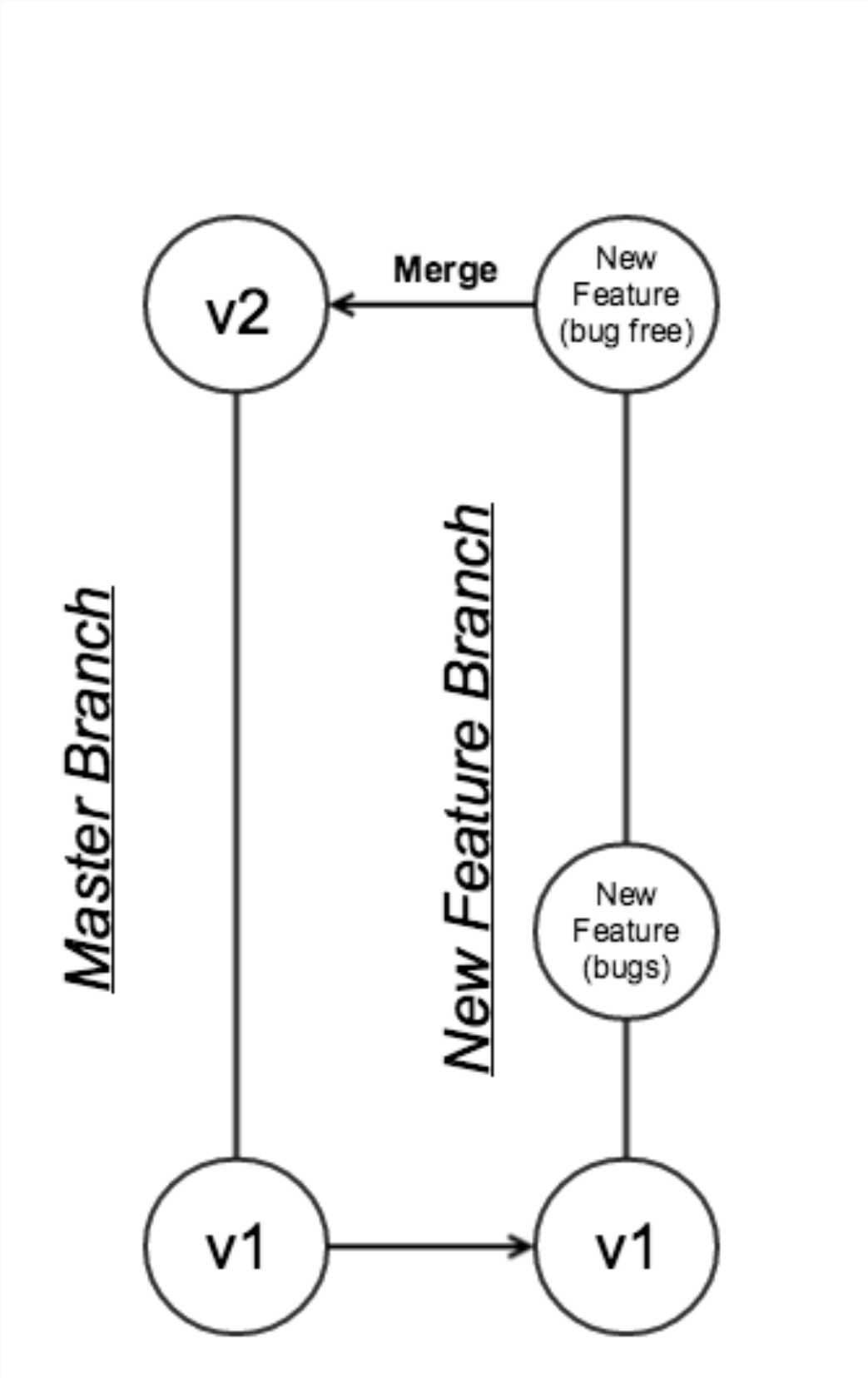
Understanding Branches



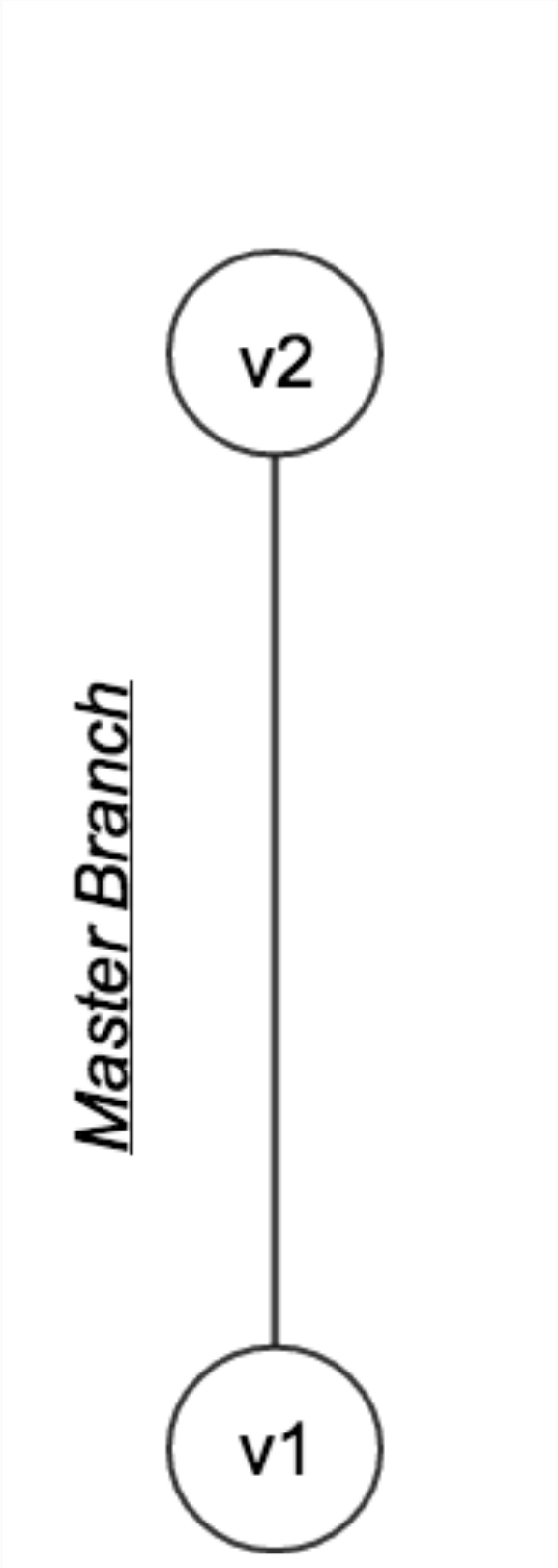
Understanding Branches



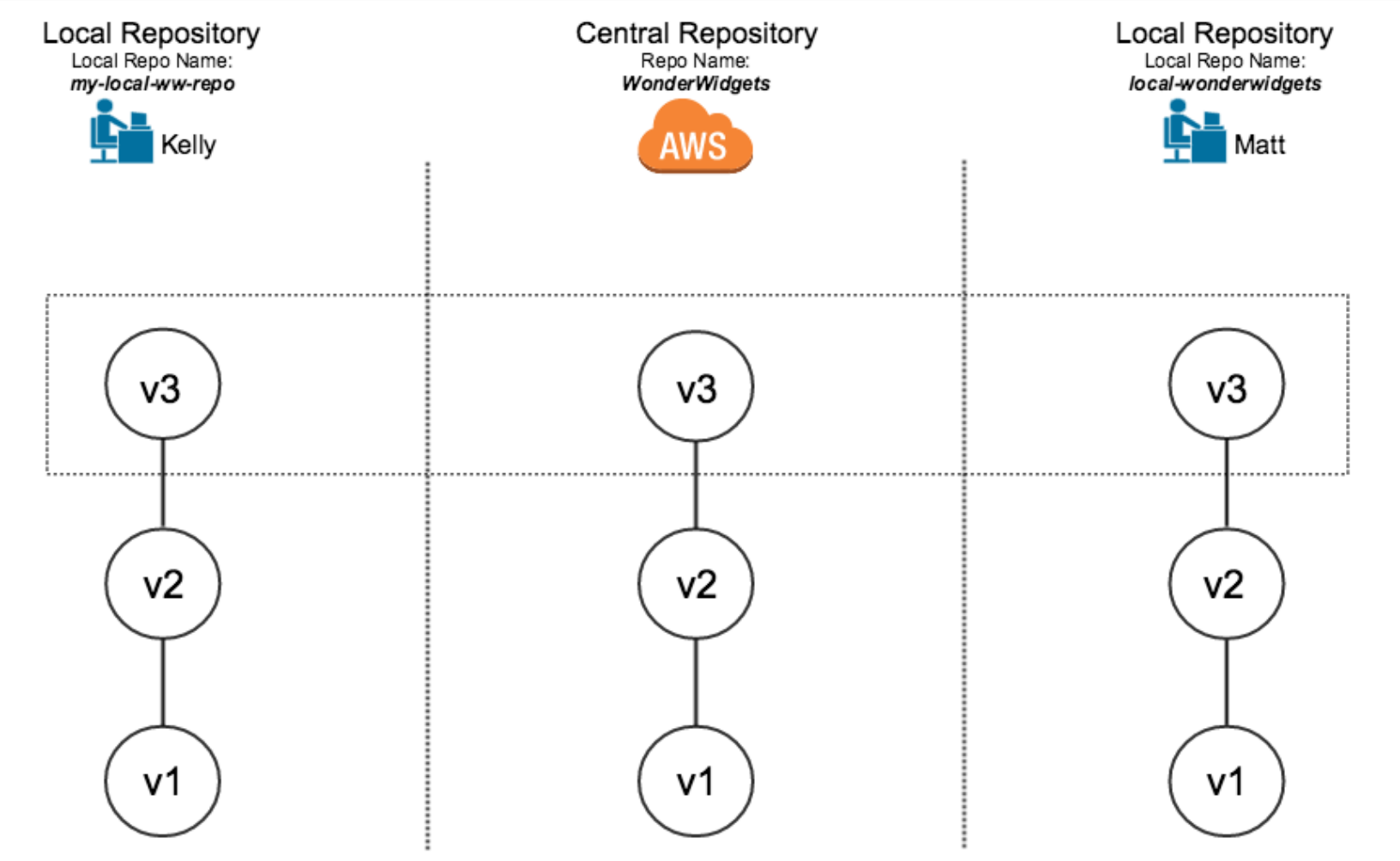
Understanding Branches



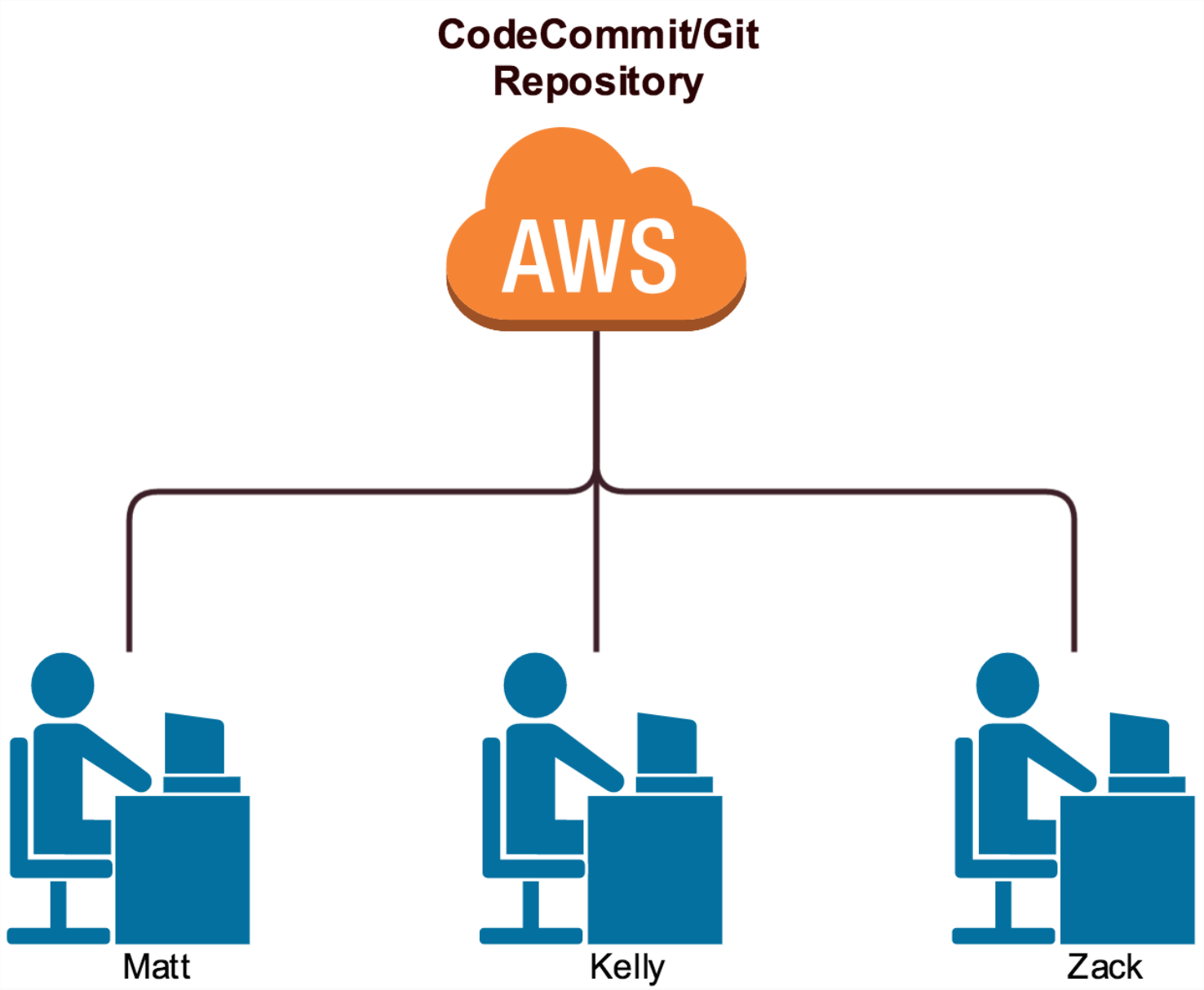
Understanding Branches



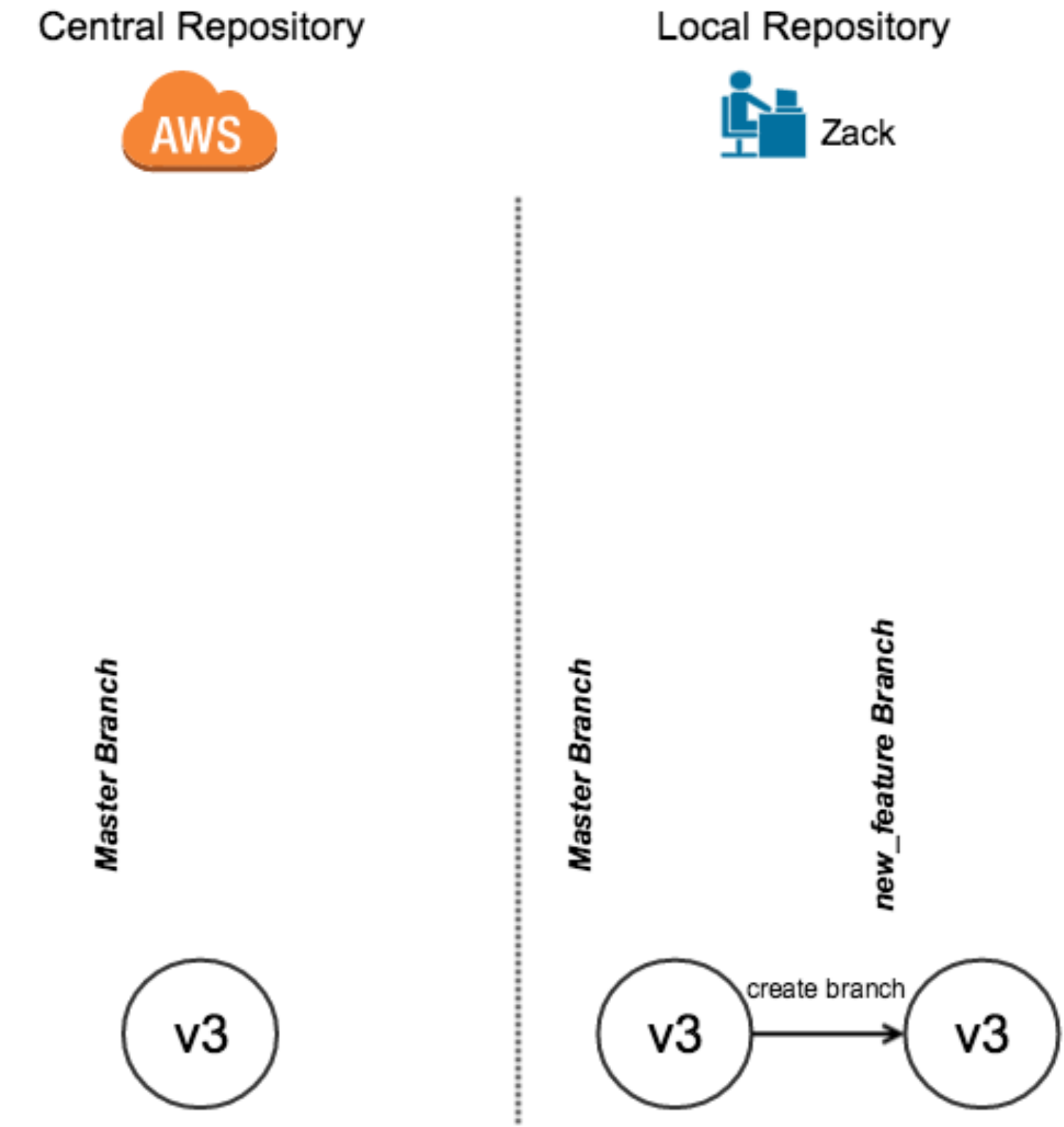
Understanding Branches



Understanding Branches



Creating a Branch



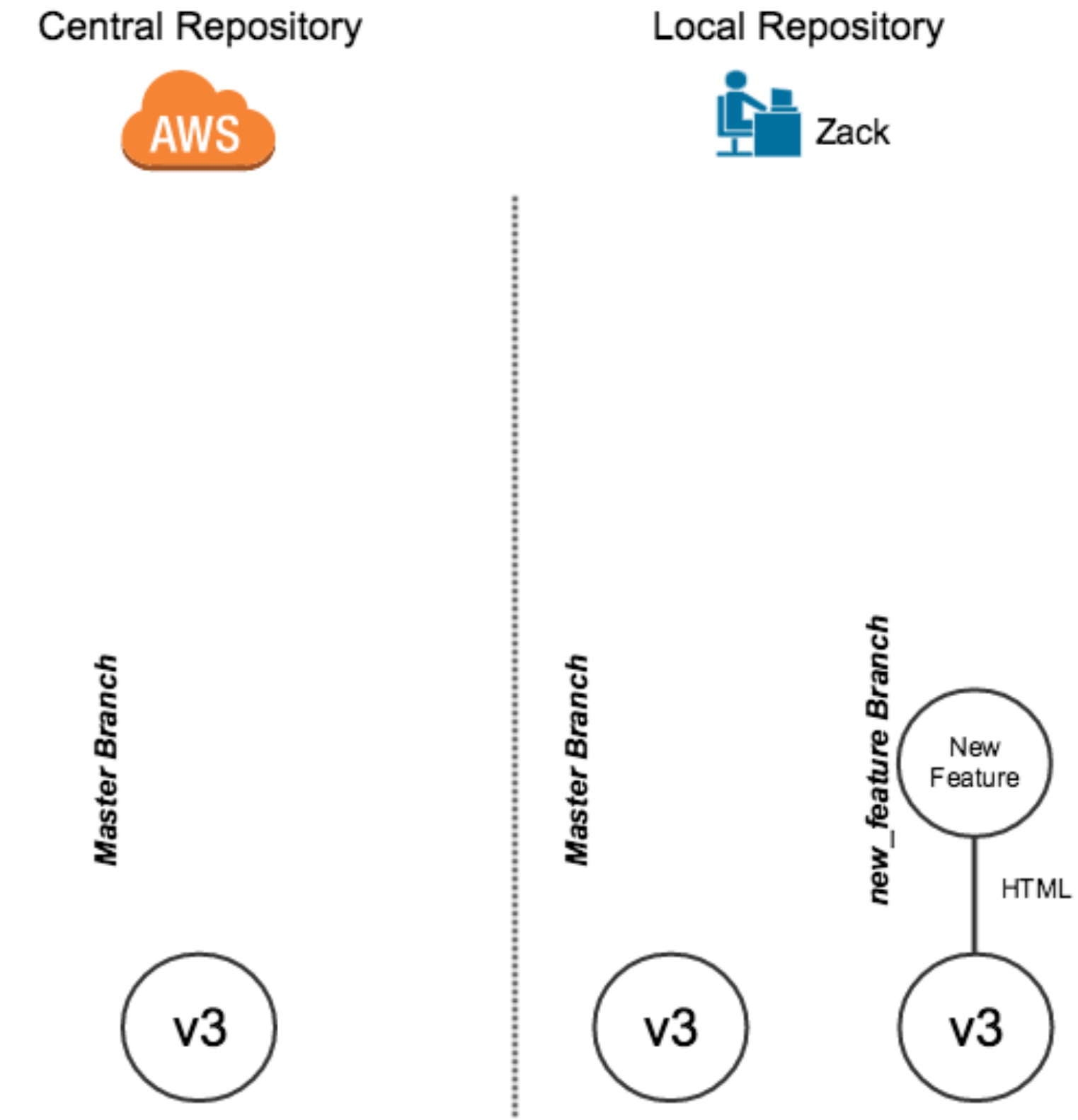
Creating a Branch

- Base Related Commands:
  git branch *(creates a new branch)*
  git checkout -b *(creates a new branch and switches you into it)*

- Base Required Attributes & Syntax:
  git branch <new branch name>
  git checkout -b <new branch name>

- Examples:
  git branch new_feature
  git checkout -b new_feature

Create the New Feature



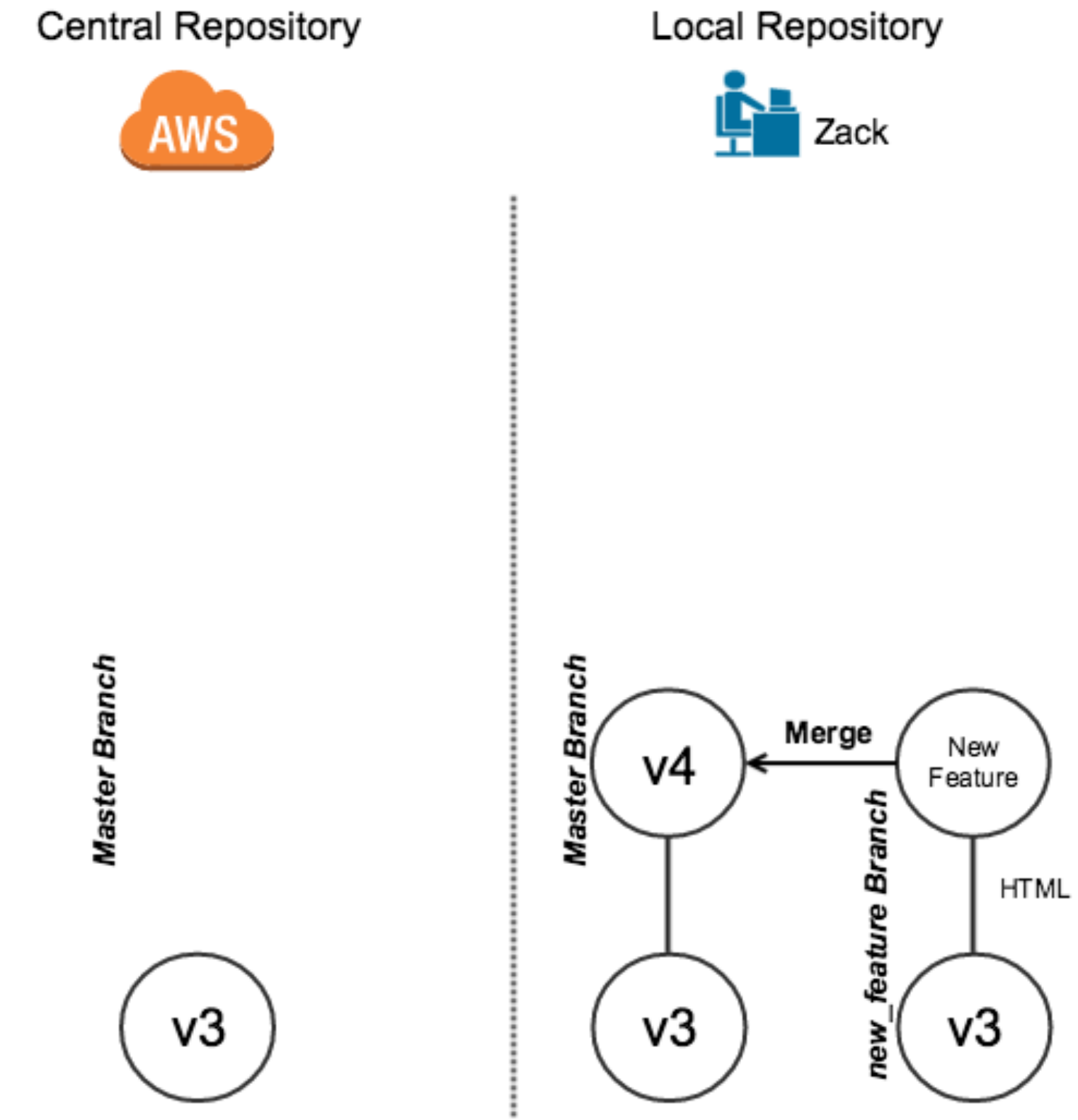
View/Change Branch

- Base Related Commands:
  git branch *(list local branches)*
  git branch -r *(list your remote branches)*
  git checkout *(switch to another branch)*

- Base Required Attributes & Syntax:
  git checkout <new branch name>

- Examples:
  git branch
  git branch -r
  git checkout master

Merge a Branch



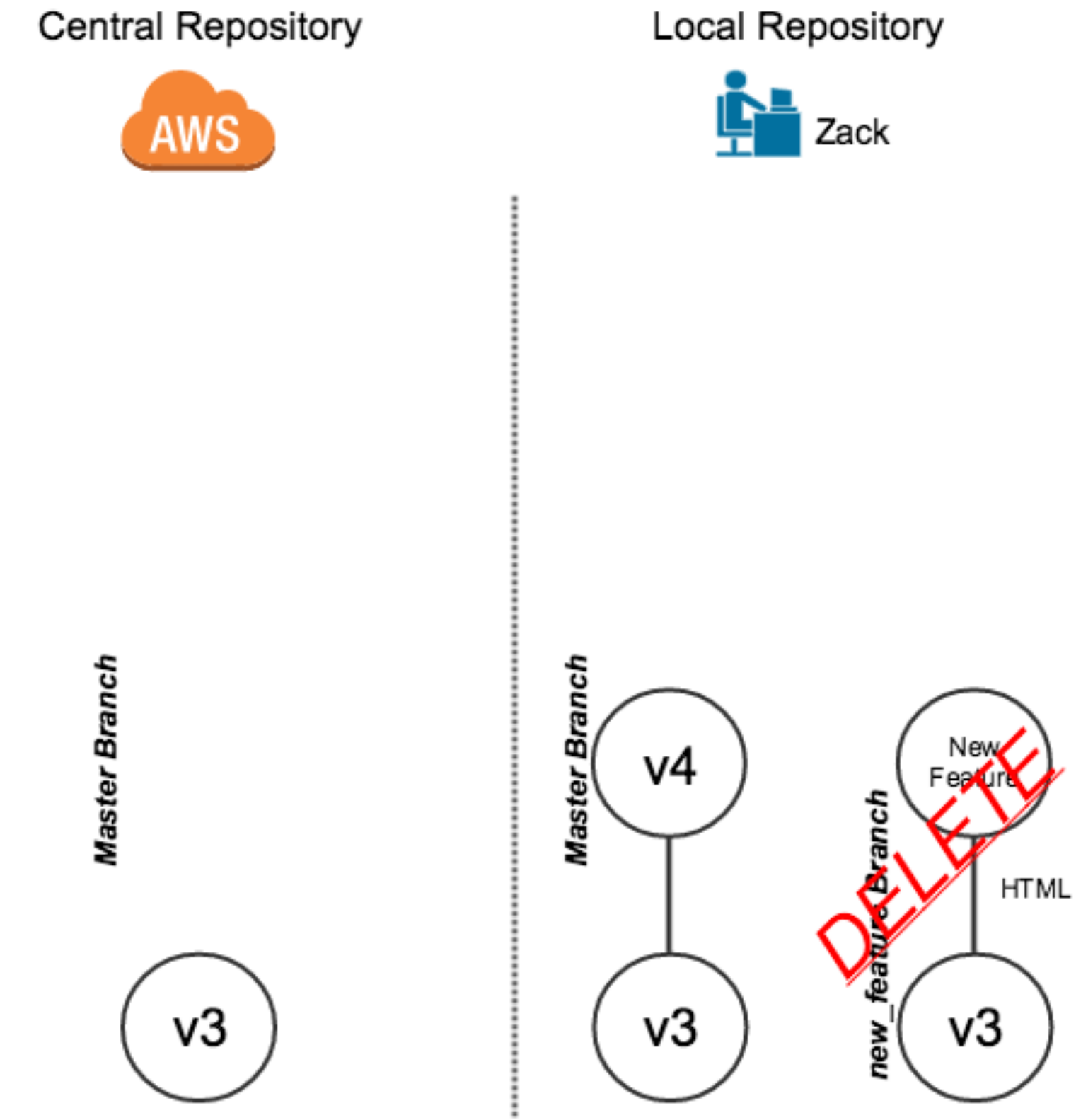
Merge a Branch

- Base Related Command:
  git merge

- Base Required Attributes & Syntax:
  git merge <branch name to merge>

- Example:
  git merge new_feature

Delete a Branch



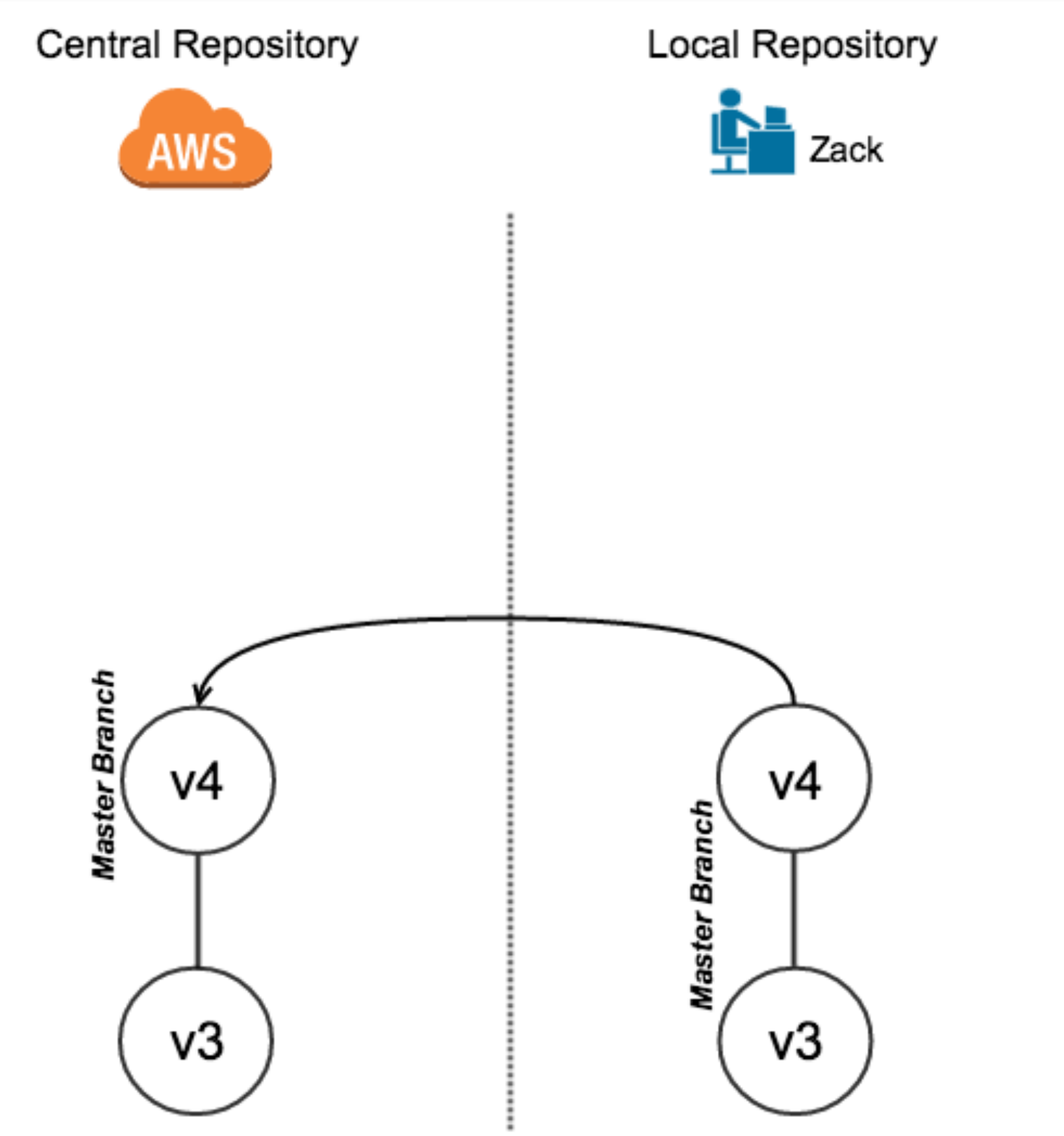
Delete a Branch

- Base Related Command:
  git branch -d (deletes a branch in the local repo unless it contains work that has not been merged)

- Base Required Attributes & Syntax:
  git branch -d <branch name>

- Example:
  git branch -d new_feature

Delete a Branch



Linux Academy

Amazon Web Services

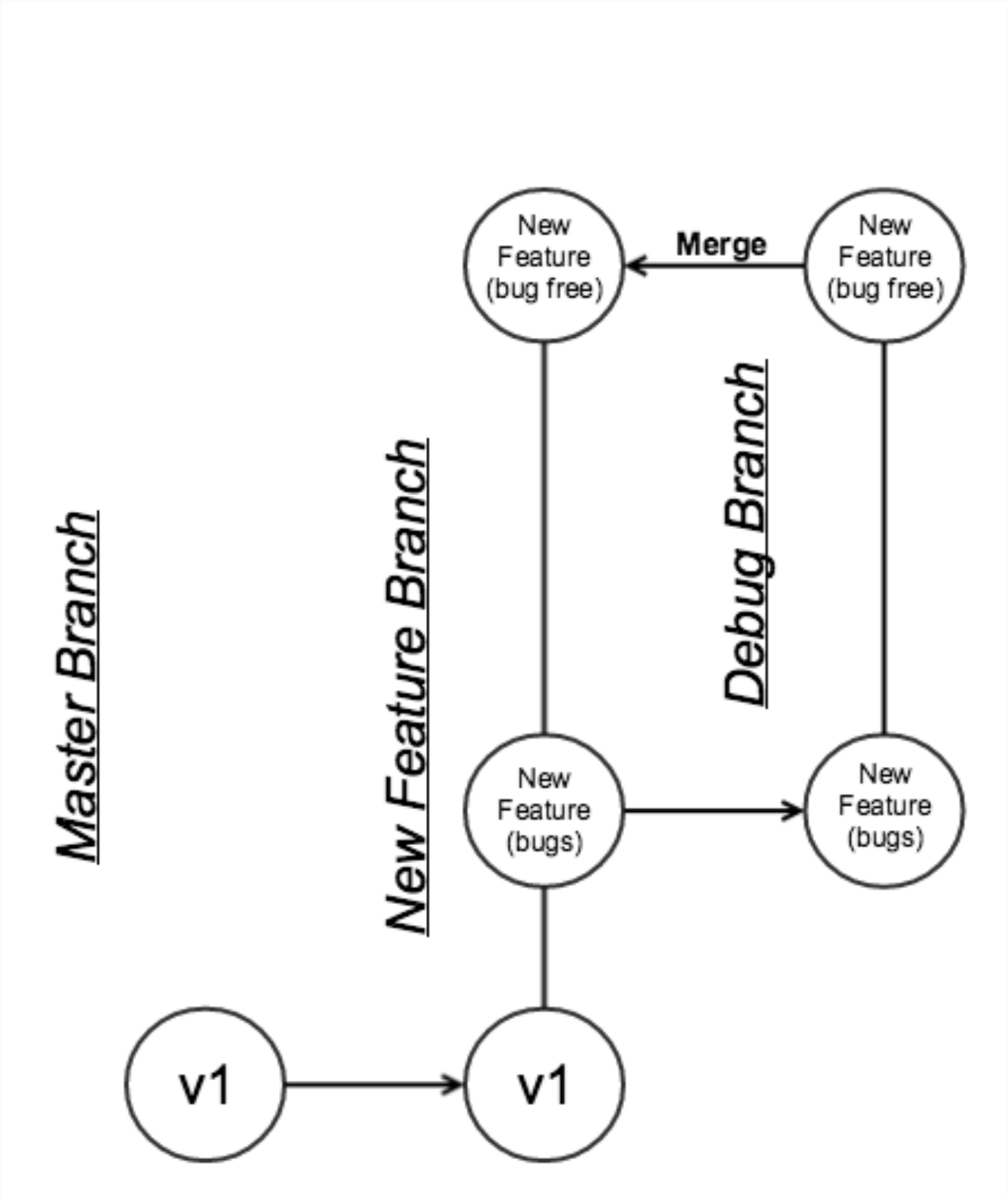
CodeCommit Basics:
Local Branches

Thank you for watching!

Amazon Web Services

CodeCommit Basics:
Commits & the HEAD Pointer

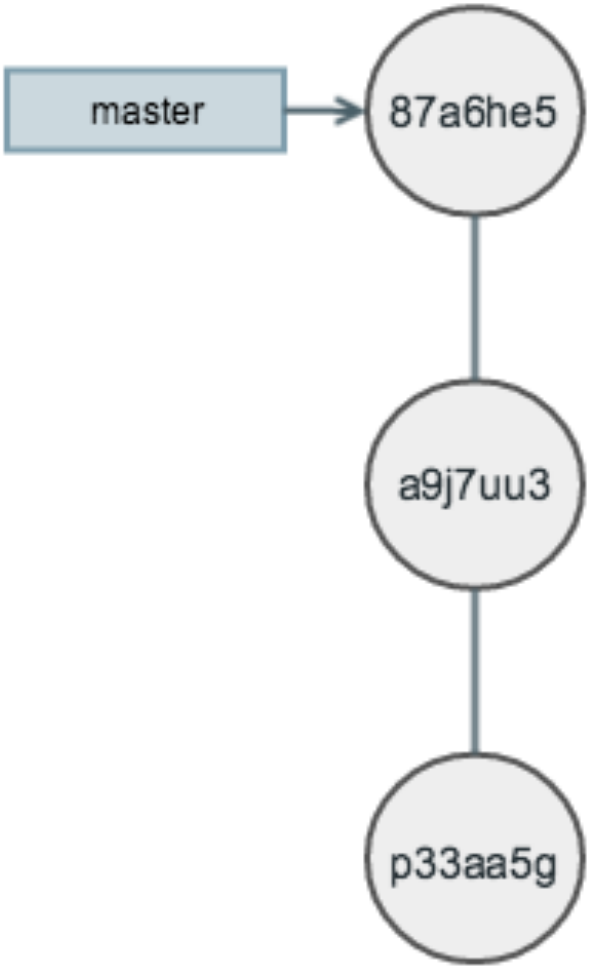
Understanding Branches



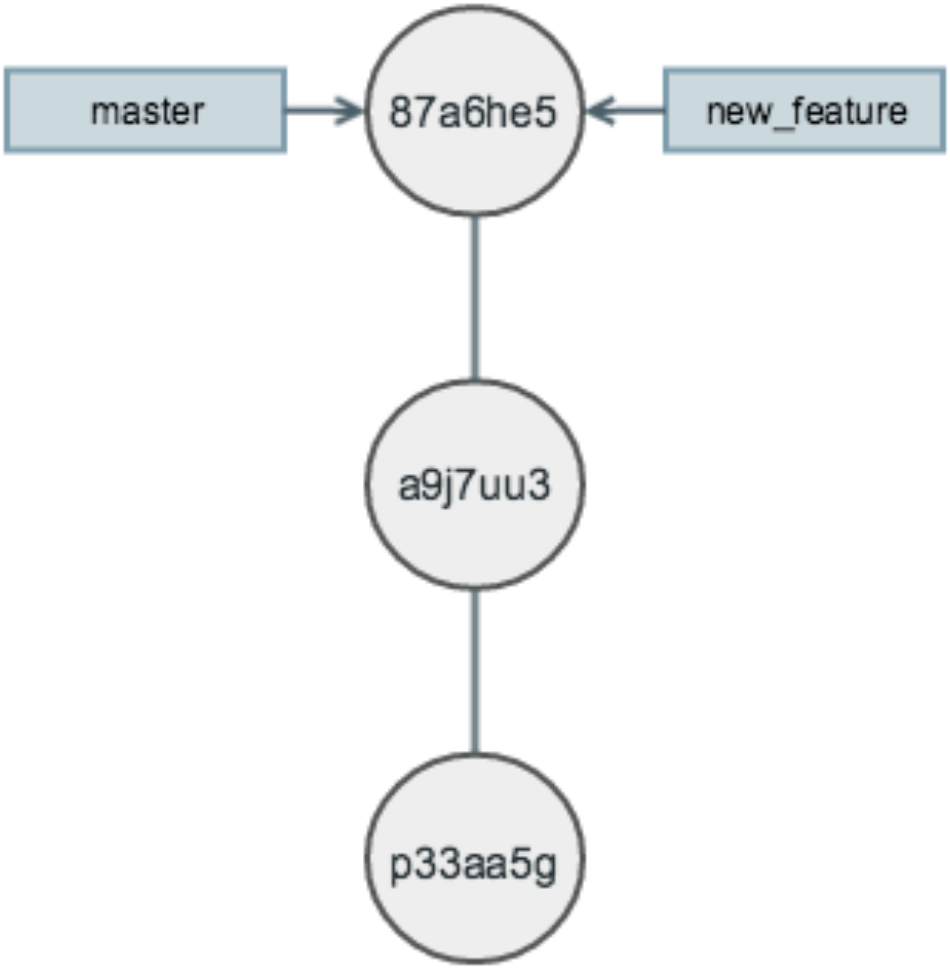
Commits & the HEAD Pointer

Commits & the HEAD Pointer

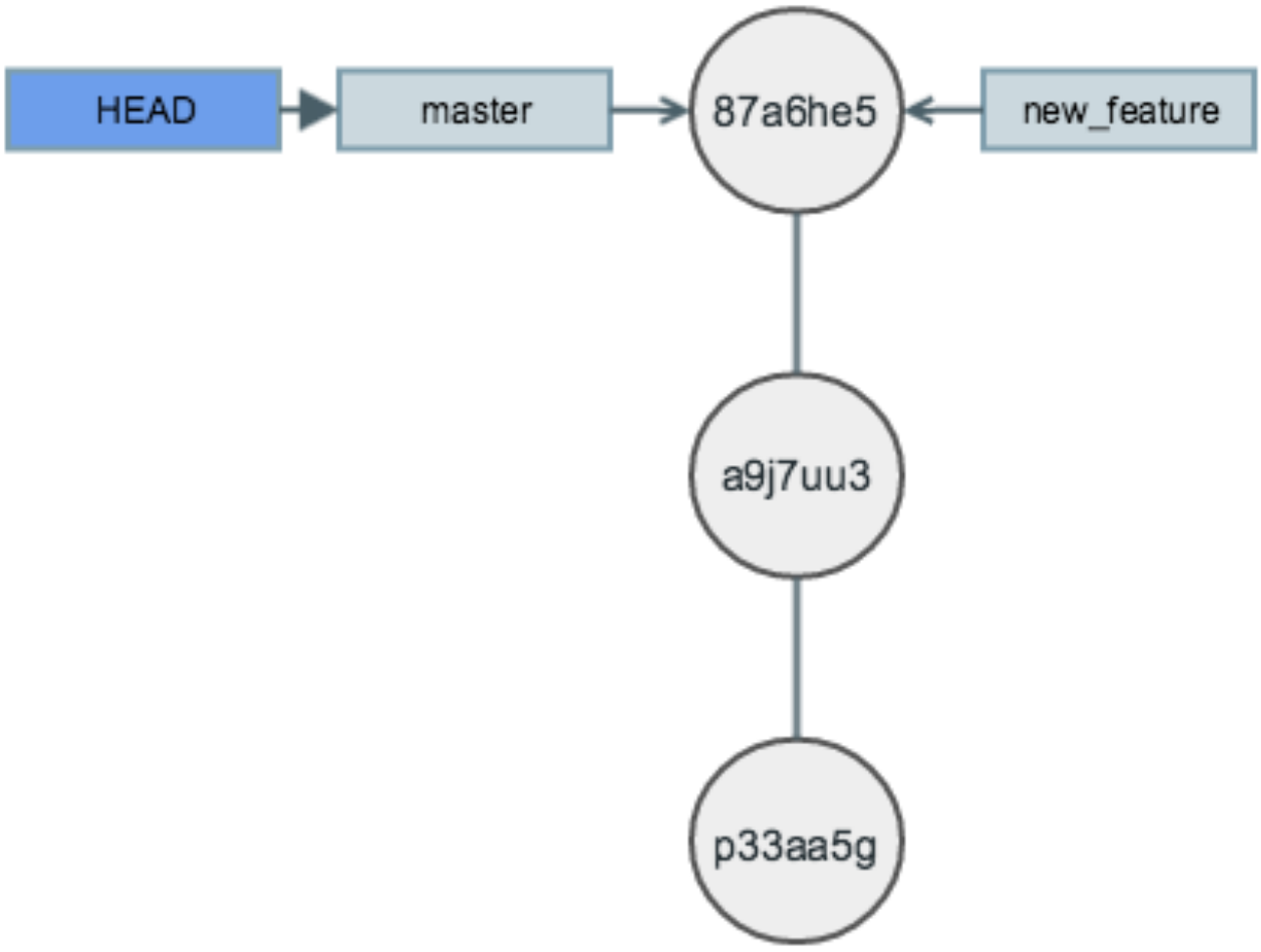
Commits & the HEAD Pointer



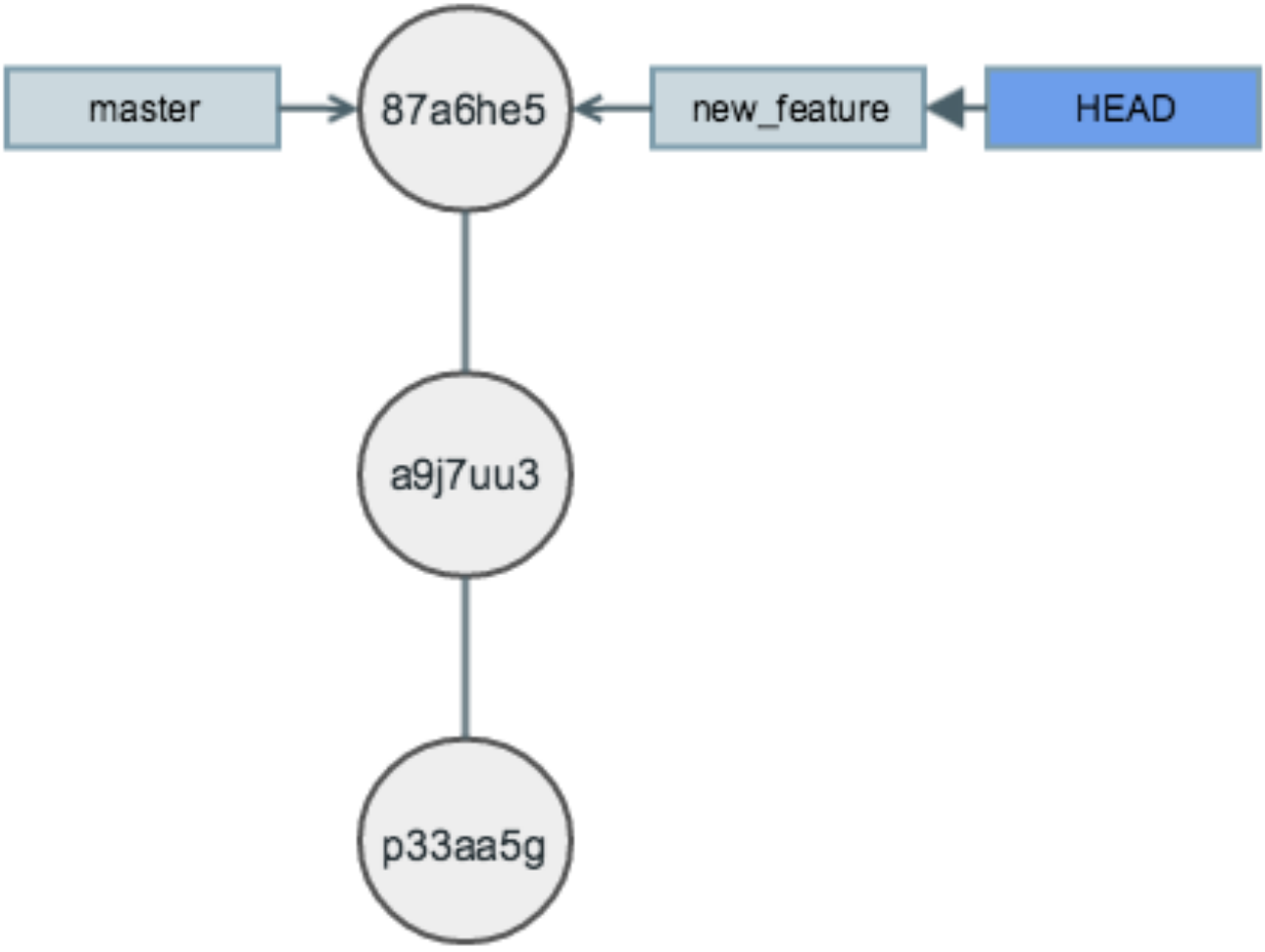
Commits & the HEAD Pointer



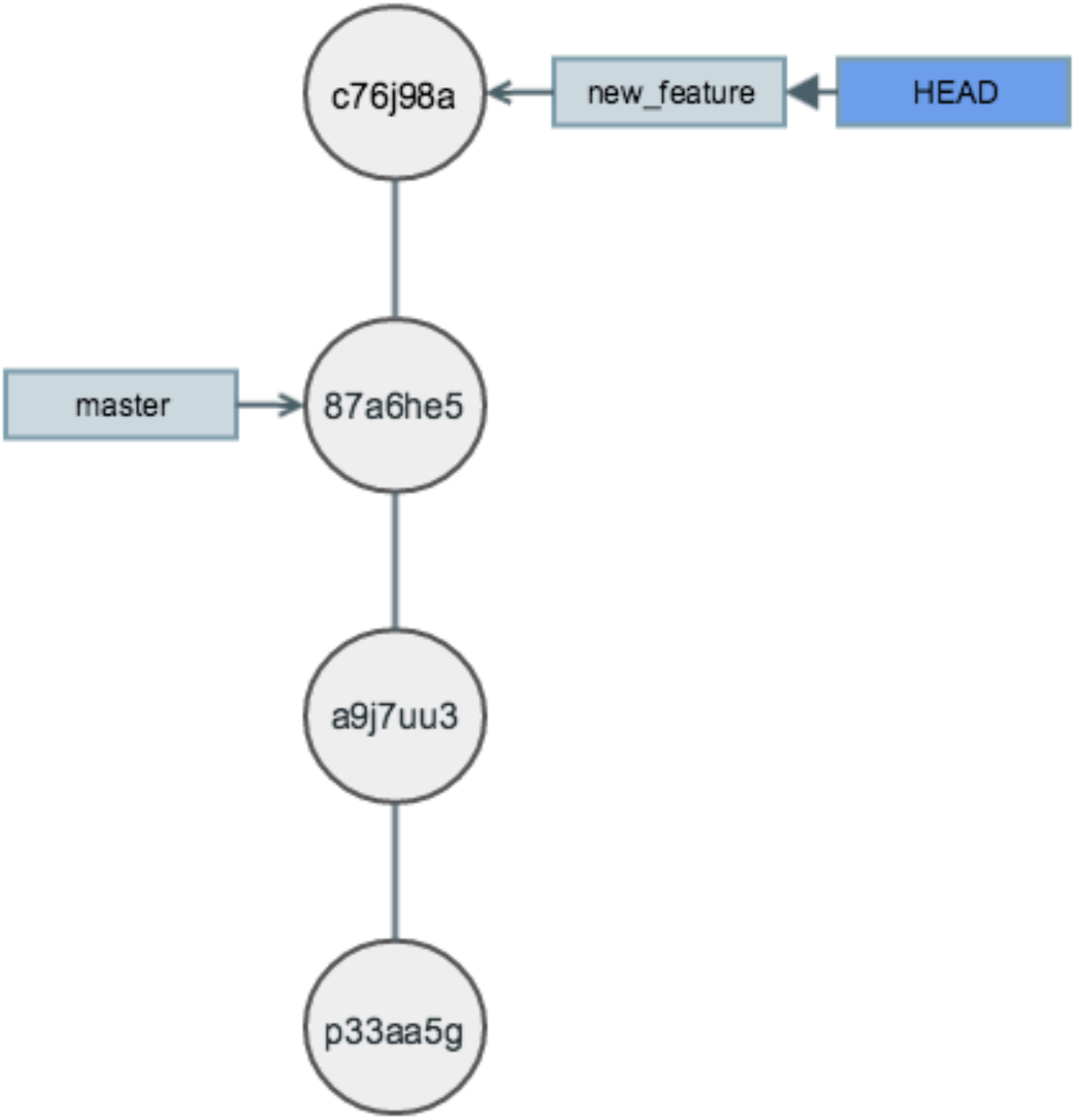
Commits & the HEAD Pointer



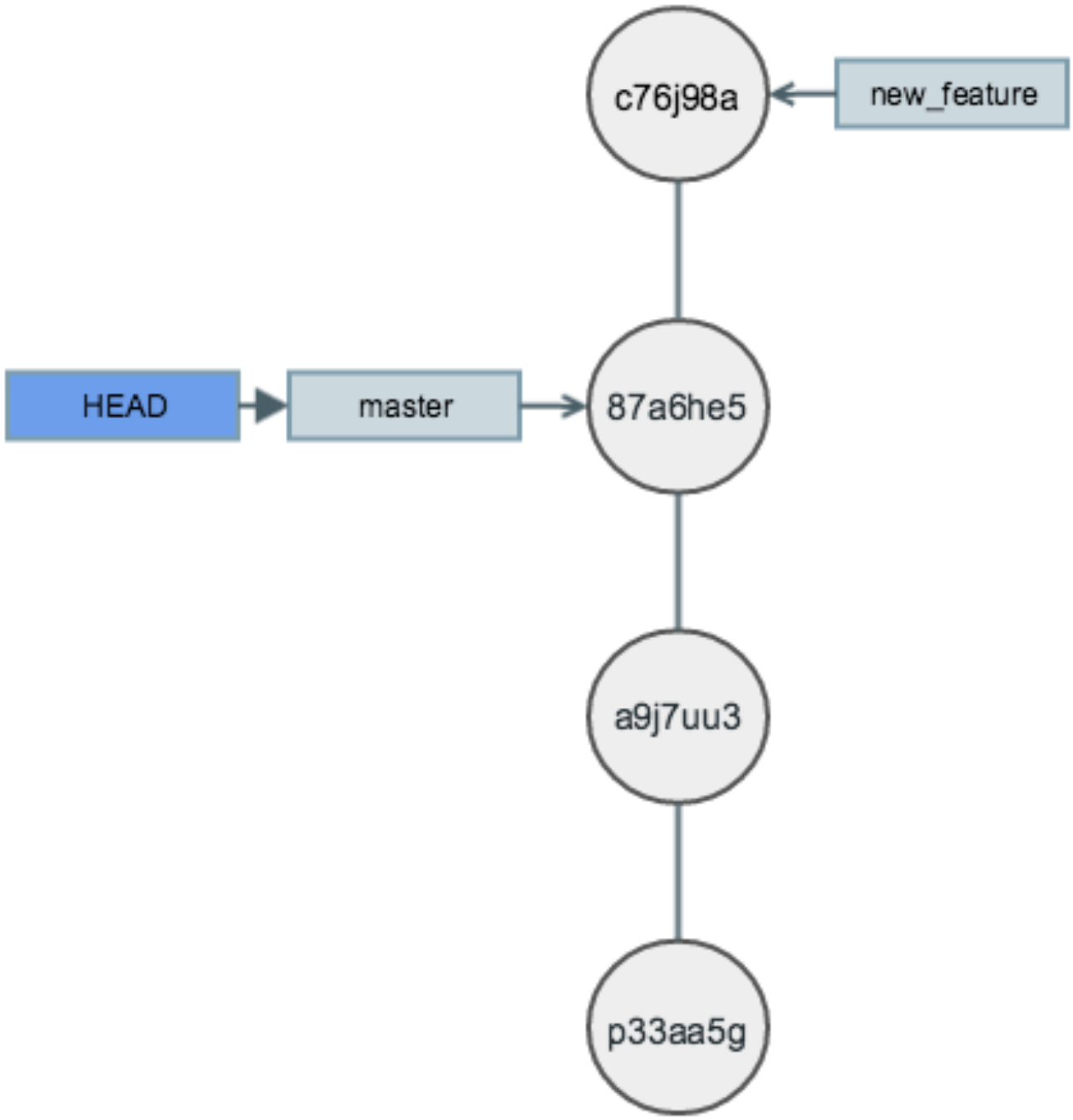
Commits & the HEAD Pointer



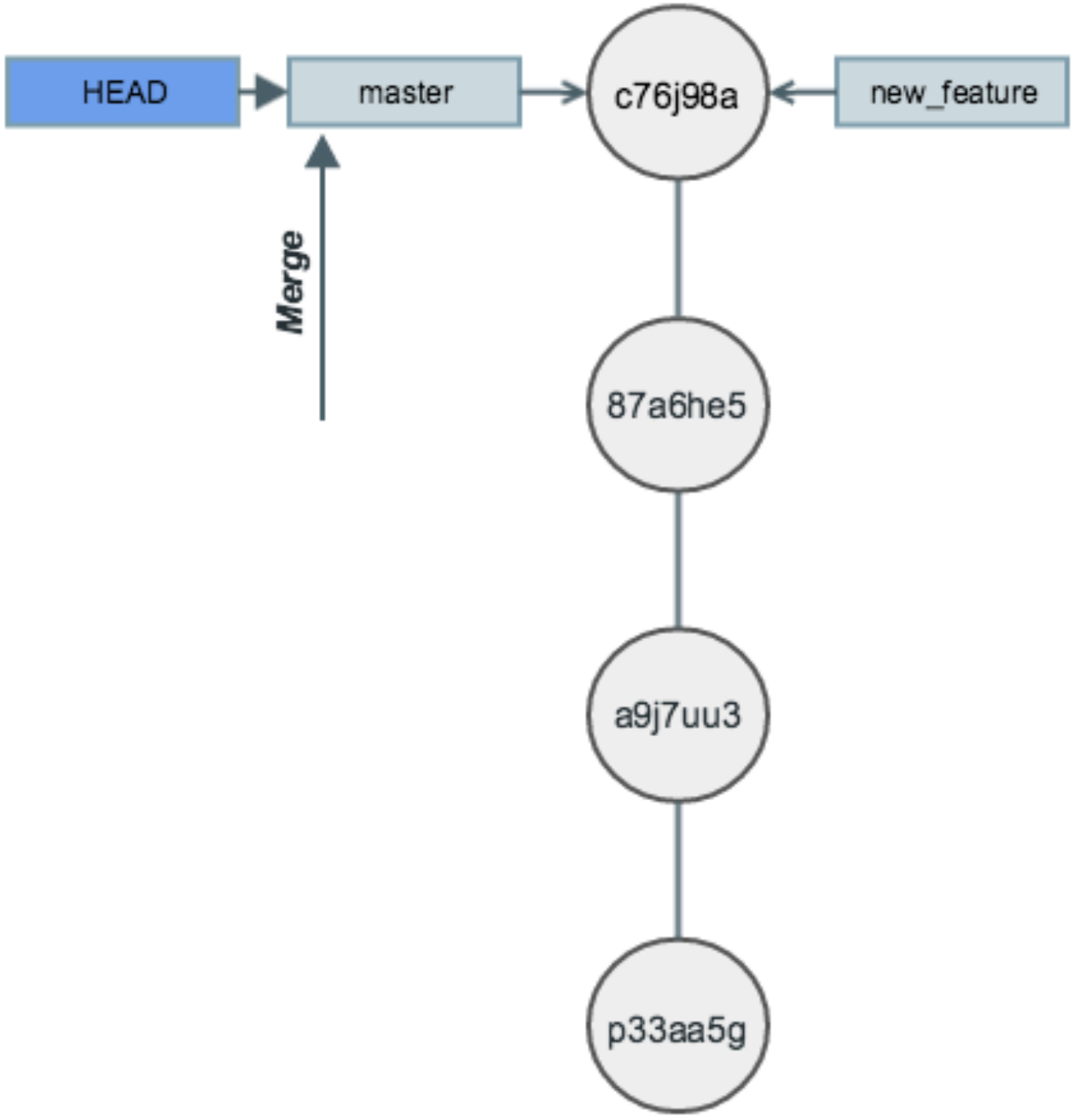
Commits & the HEAD Pointer



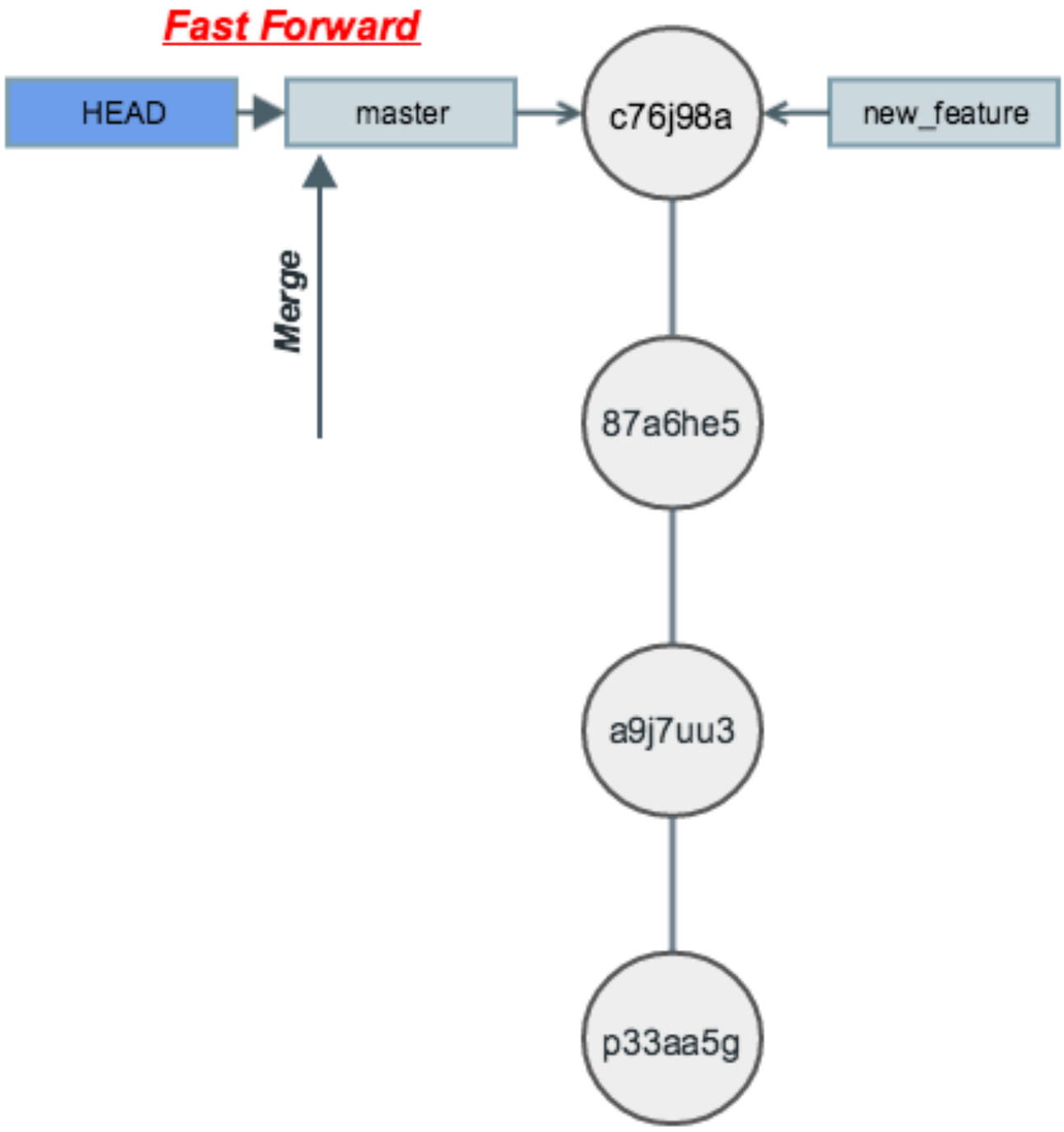
Commits & the HEAD Pointer



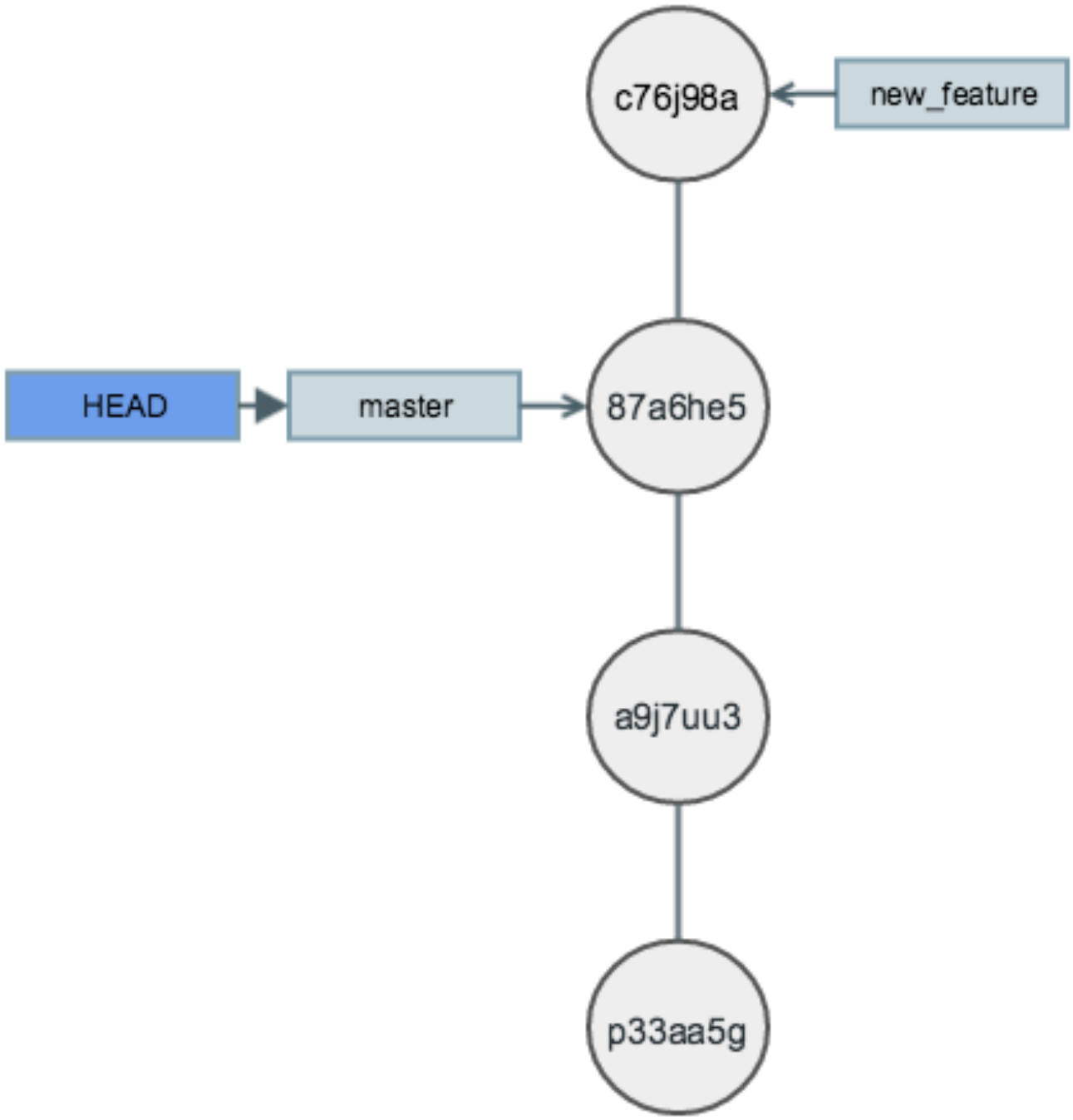
Commits & the HEAD Pointer



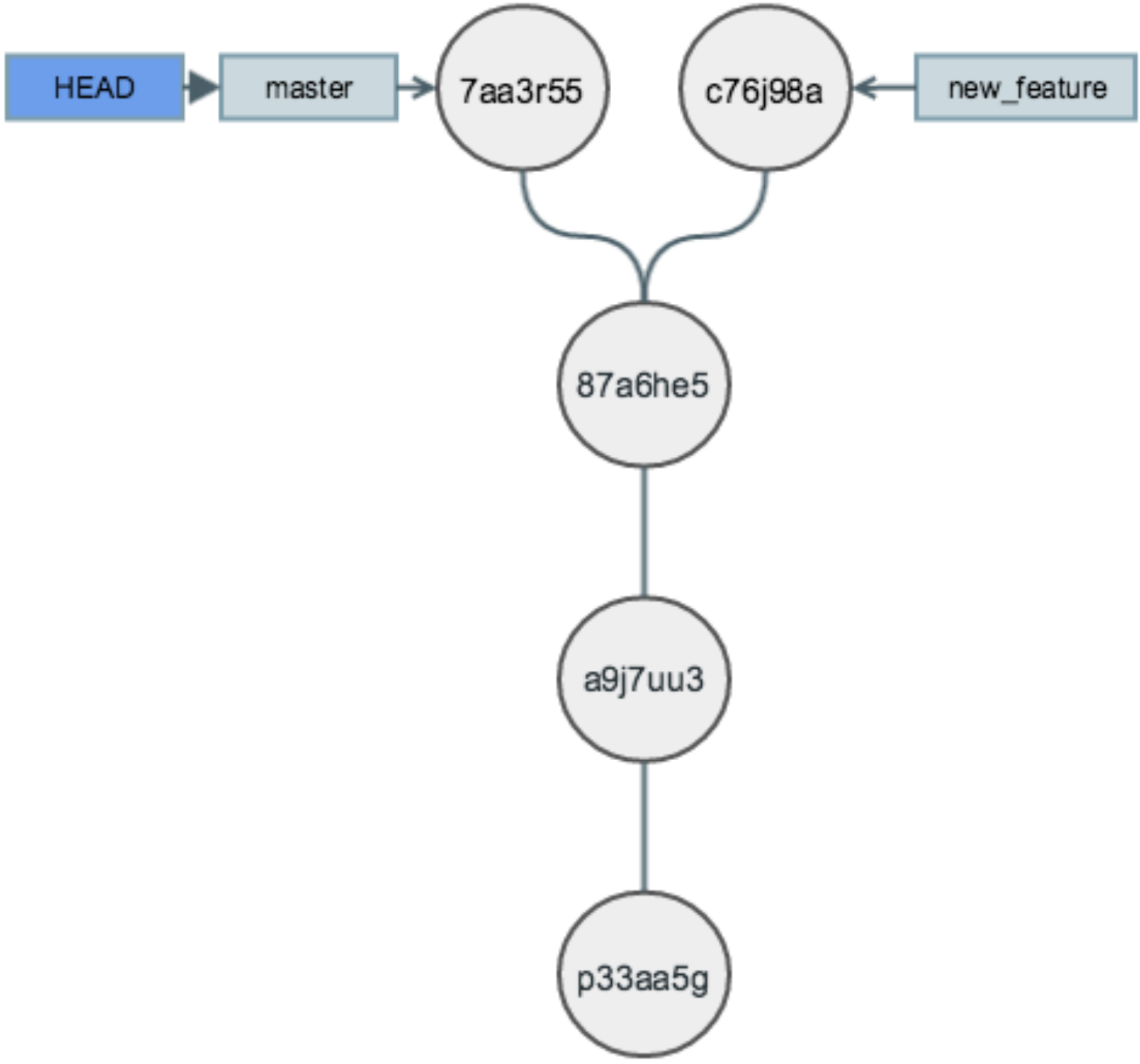
Commits & the HEAD Pointer



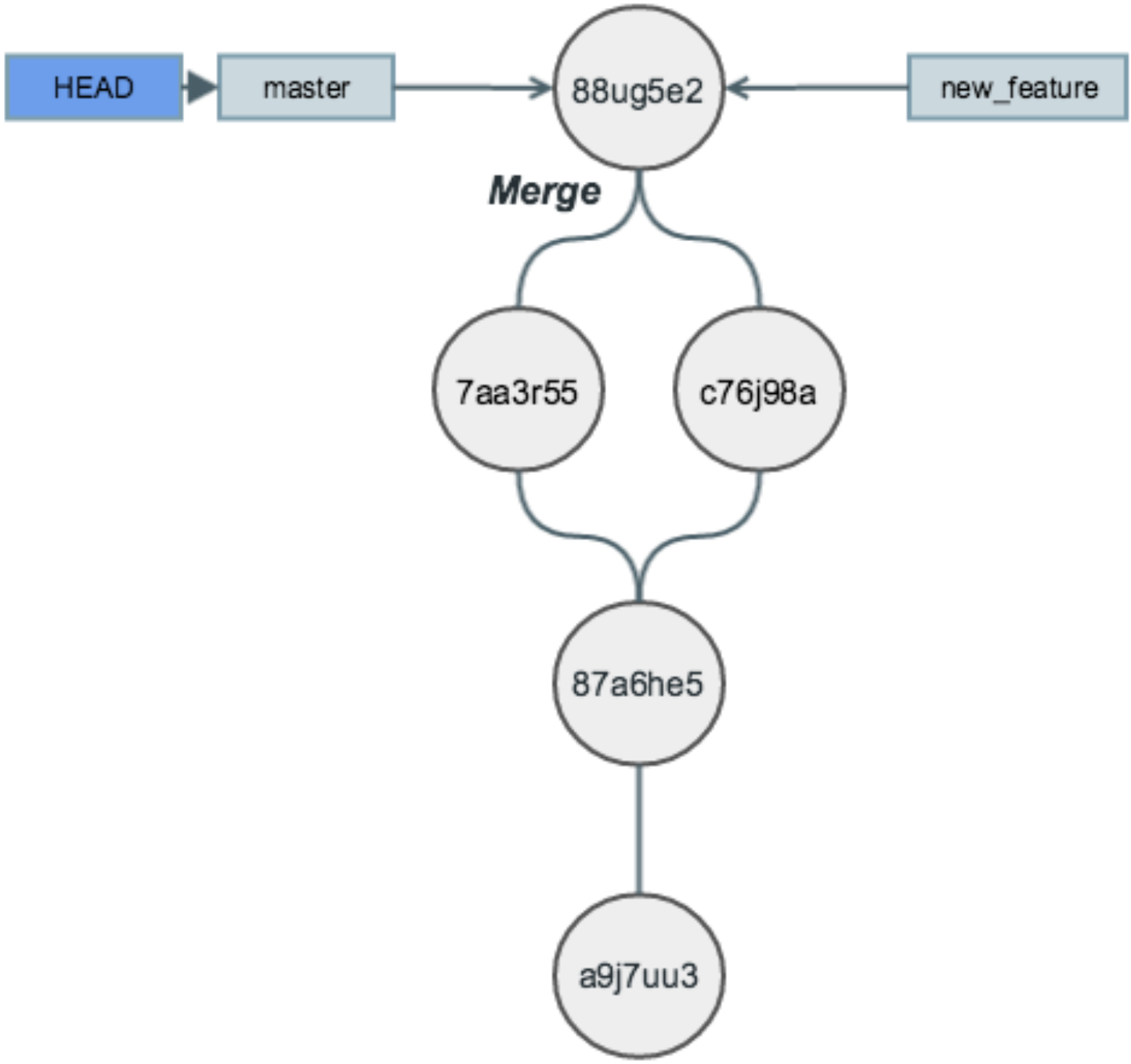
Commits & the HEAD Pointer



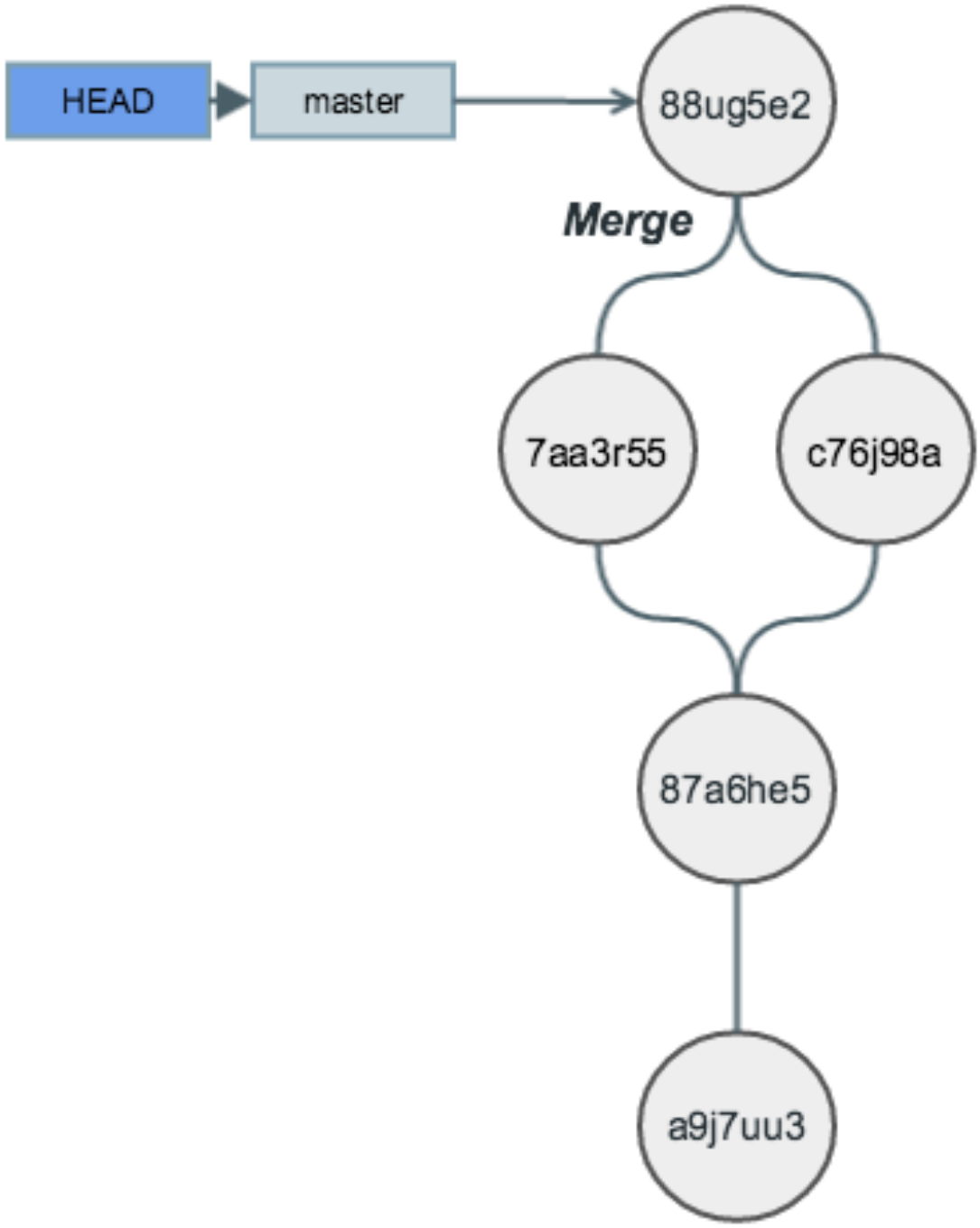
Commits & the HEAD Pointer



Commits & the HEAD Pointer



Commits & the HEAD Pointer



Linux Academy

Amazon Web Services

CodeCommit Basics:
Commits & the HEAD Pointer

Thank you for watching!

Amazon Web Services

CodeCommit Basics: Tags

Understanding Tags

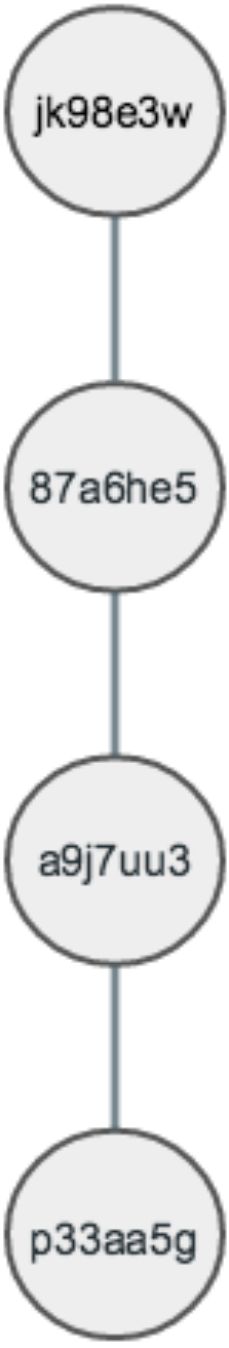
- Definition:
  A tag is a symbolic name, generally given to a commit, that identifies it as having some importance.

- What are tags used for?:
  Tags are generally used to mark release points (i.e. v1.0, or v2.0), OR a version released to a specific customer (i.e. v1.2 released to company XYZ).

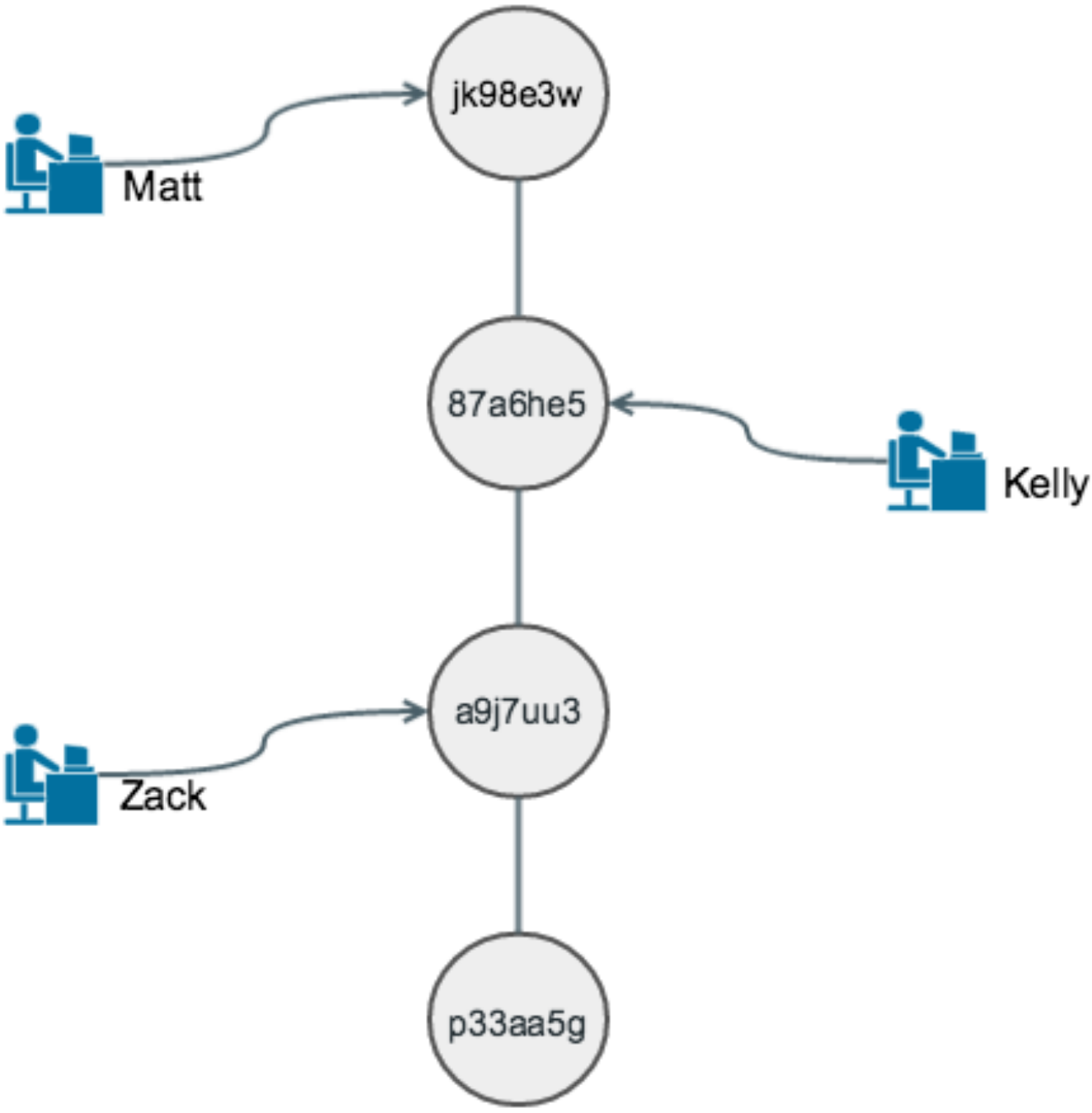
- Tags vs. Branches:
  Tags and branches work together to assist developers with version control. While branches are pointers that can "move" and point to different commits, a tag is an identifier that always points to the same commit.

Think of a tag as a post-it-note that you attached to a commit

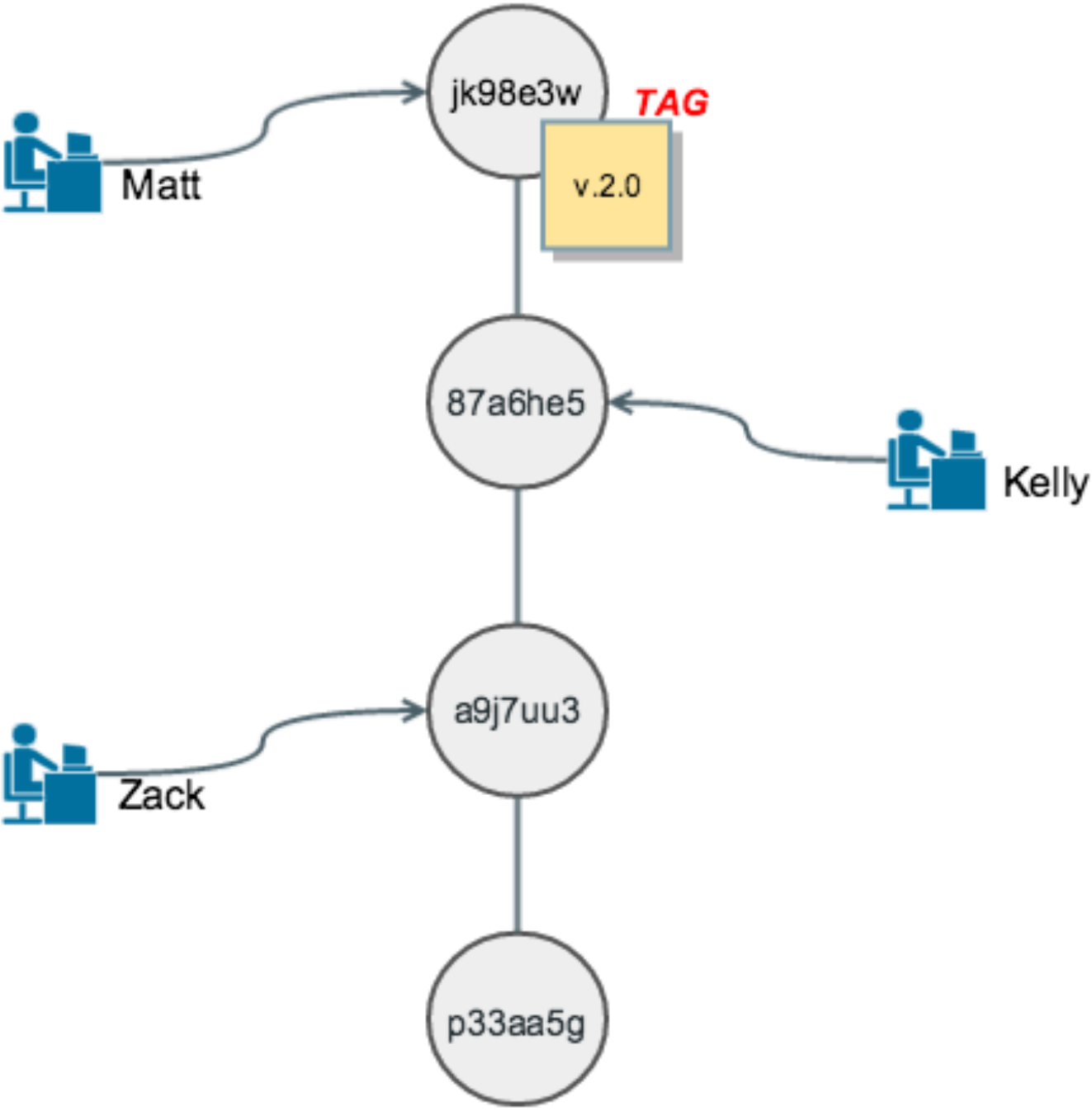
Understanding Tags



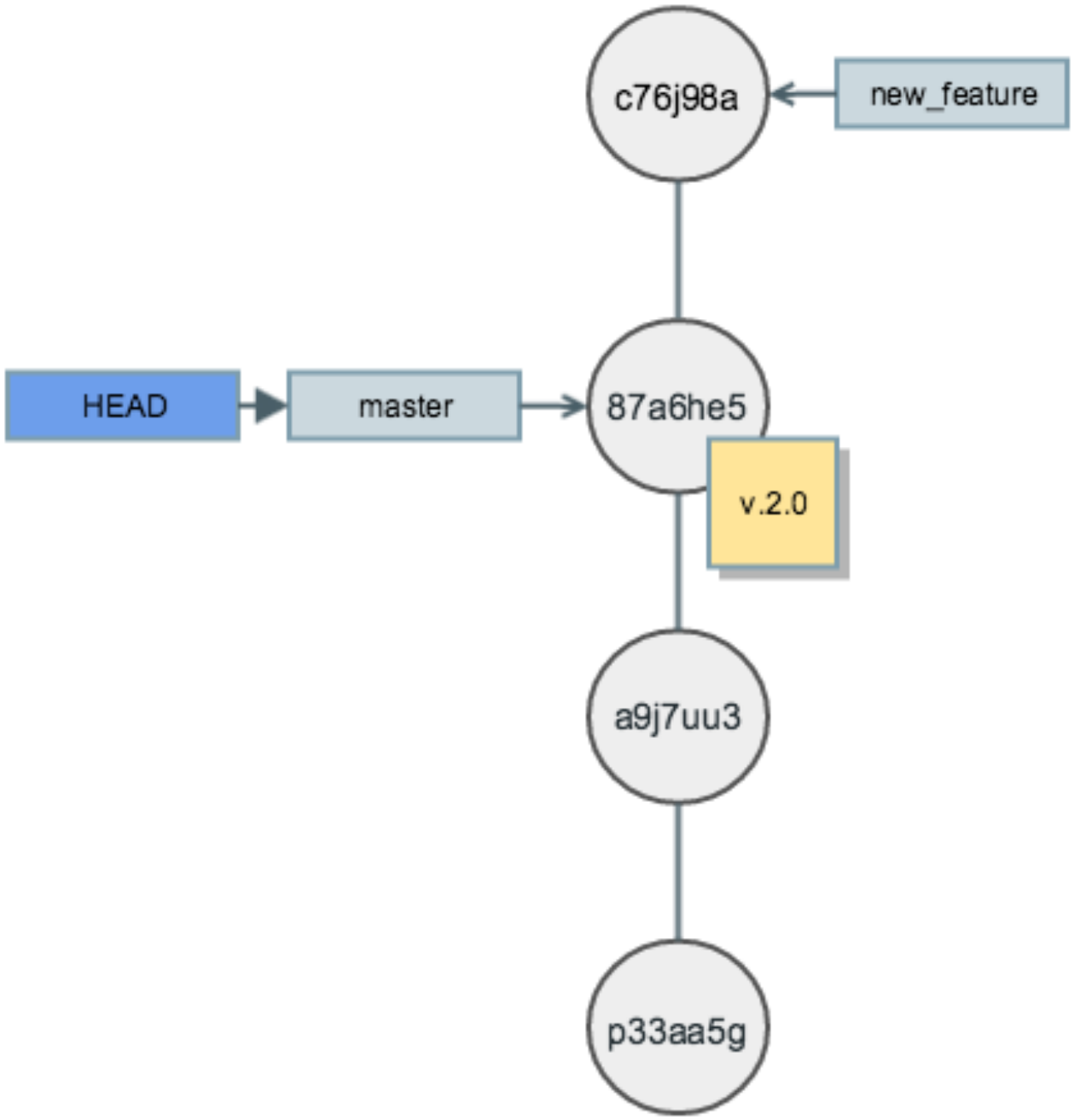
Understanding Tags



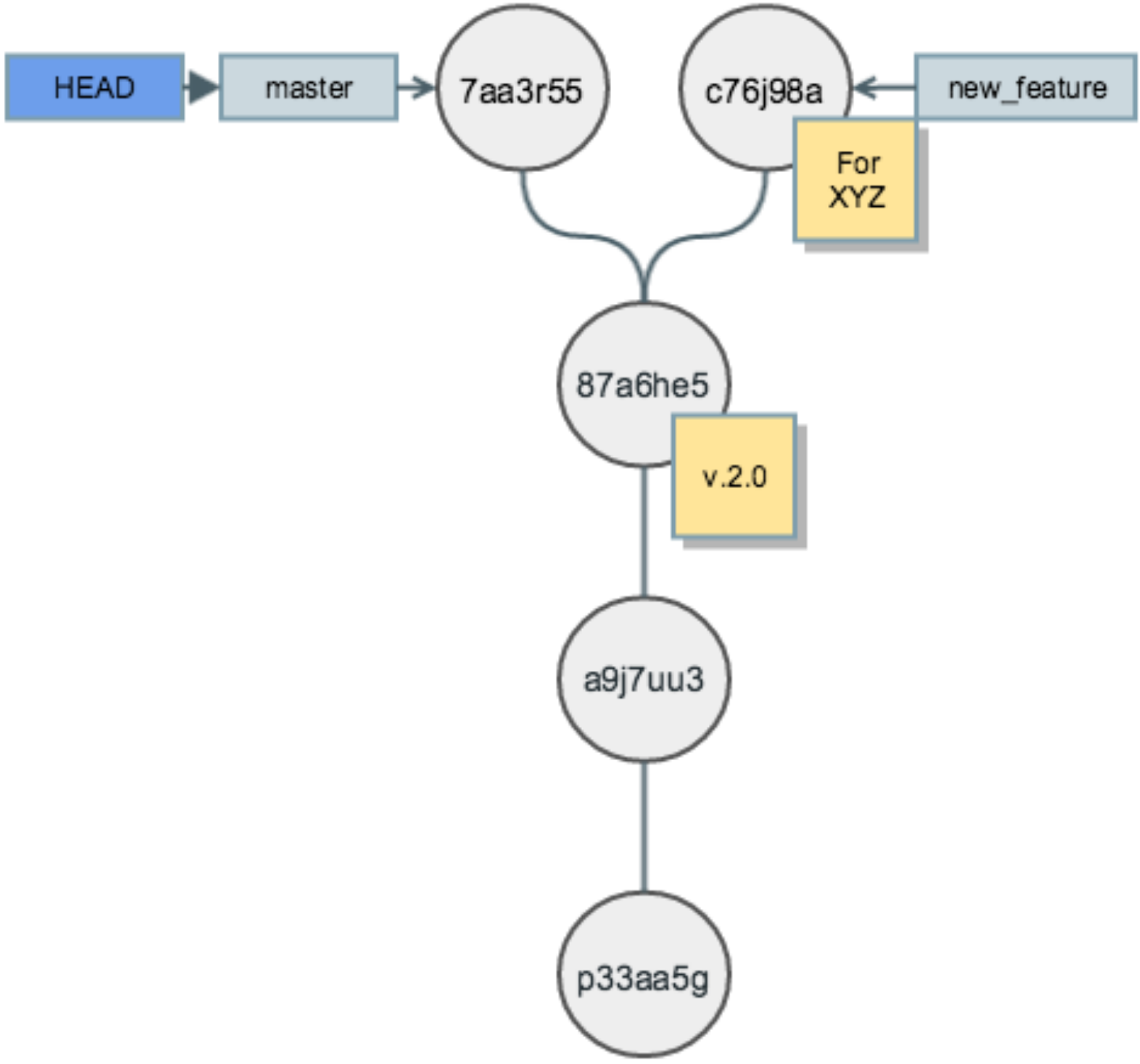
Understanding Tags



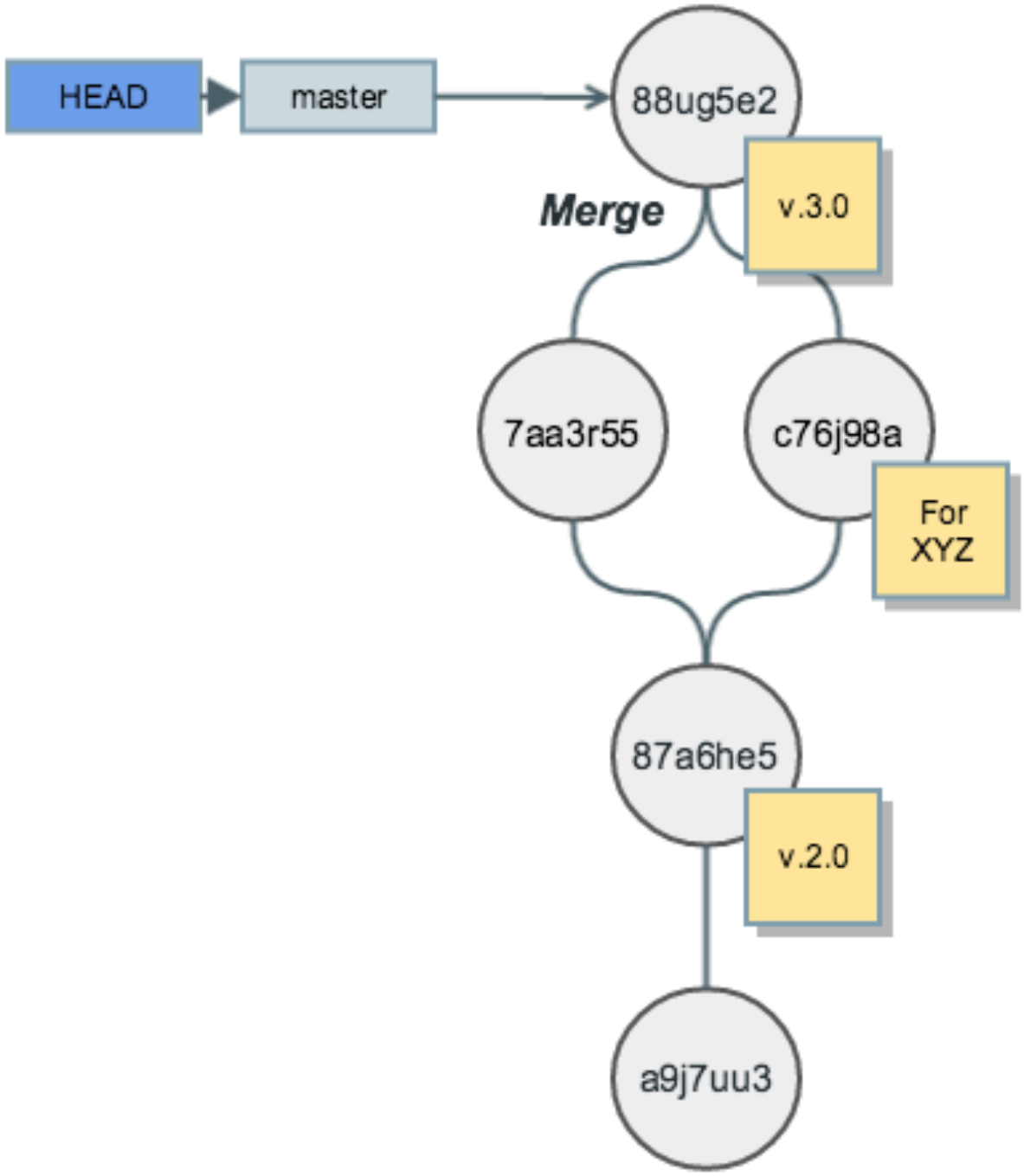
Understanding Tags



Understanding Tags



Understanding Tags



Create, View & Delete Tags

- Base Related Command:
  git tag *(used to create/list/delete tags)*
  git show *(view info about a specific tag)*

- Based Required Attributes & Syntax:
  git tag <tag name> <commit id#> *(create a tag)*
  git tag *(list tags)*
  git tag -d <tag name> *(delete a tag)*
  git show <tag name> *(detailed tag info)*

- Examples:
  git tag v1.0 dc08f3a
  git tag
  git tag -d v1.0
  git show v1.0

Managing Tags with the Central Repo

- Base Related Command:
  - ***git push*** *(push to or delete tag from the central repo)*
  - ***git ls-remote*** *(view tags in the central repo)*
  - ***git fetch*** *(pull tags from the central repo)*

- Based Required Attributes & Syntax:
  - ***git push <remote name> <tag name>*** *(add)*
  - ***git ls-remote --tags***
  - ***git fetch --tags***
  - ***git push --delete <remote name> <tag name>*** *(delete)*

- Examples:
  - ***git push origin v1.0***
  - ***git ls-remote --tags***
  - ***git fetch --tags***
  - ***git push --delete origin v1.0***

Linux Academy

Amazon Web Services

CodeCommit Basics: Tags

Thank you for watching!

Amazon Web Services

CodeCommit Basics:
Migrating a Repository into CodeCommit

Migrating a Repository into CodeCommit

- AWS allows you to migrate other **git**-based repositories seamlessly into CodeCommit.
  - Github
  - Beanstalk
  - GitLab
  - BitBucket

- You can choose to migrate the entire repository or just some of the branches.

- You can migrate from other (non **git**-based) version control systems, such as Perforce, Subversion, or TFS, but you will have to migrate to a **git**-based system first.

- You can choose to migrate the entire repository at once, or in parts (incrementally).

Should I Migrate Incrementally?

- The repository you want to migrate has more than five years of history.
- Your internet connection is subject to intermittent outages, dropped packets, slow response, or other interruptions in service.
- The overall size of the repository is larger than 2 GB and you intend to migrate the entire repository.
- The repository contains large artifacts or binaries that do not compress well, such as large image files with more than five tracked versions.
- You have previously attempted a migration to AWS CodeCommit and received an "Internal Service Error" message.

Migrating a Repository into CodeCommit

1) Create a new repository in AWS CodeCommit

2) Clone the repo we want to migrate to our local machine (into a temp directory)

3) Push the cloned GitHub repo on our local machine to the repo we created in AWS CodeCommit during step 1

4) Delete the temporary directory that housed the cloned GitHub repo on our local machine

5) Clone the AWS CodeCommit repo that we migrated to our local machine

Linux Academy

Amazon Web Services

CodeCommit Basics:
Migrating a Repository into CodeCommit

Thank you for watching!

Amazon Web Services

CodeCommit & Other AWS Services: Introduction to Triggers

What are Triggers?

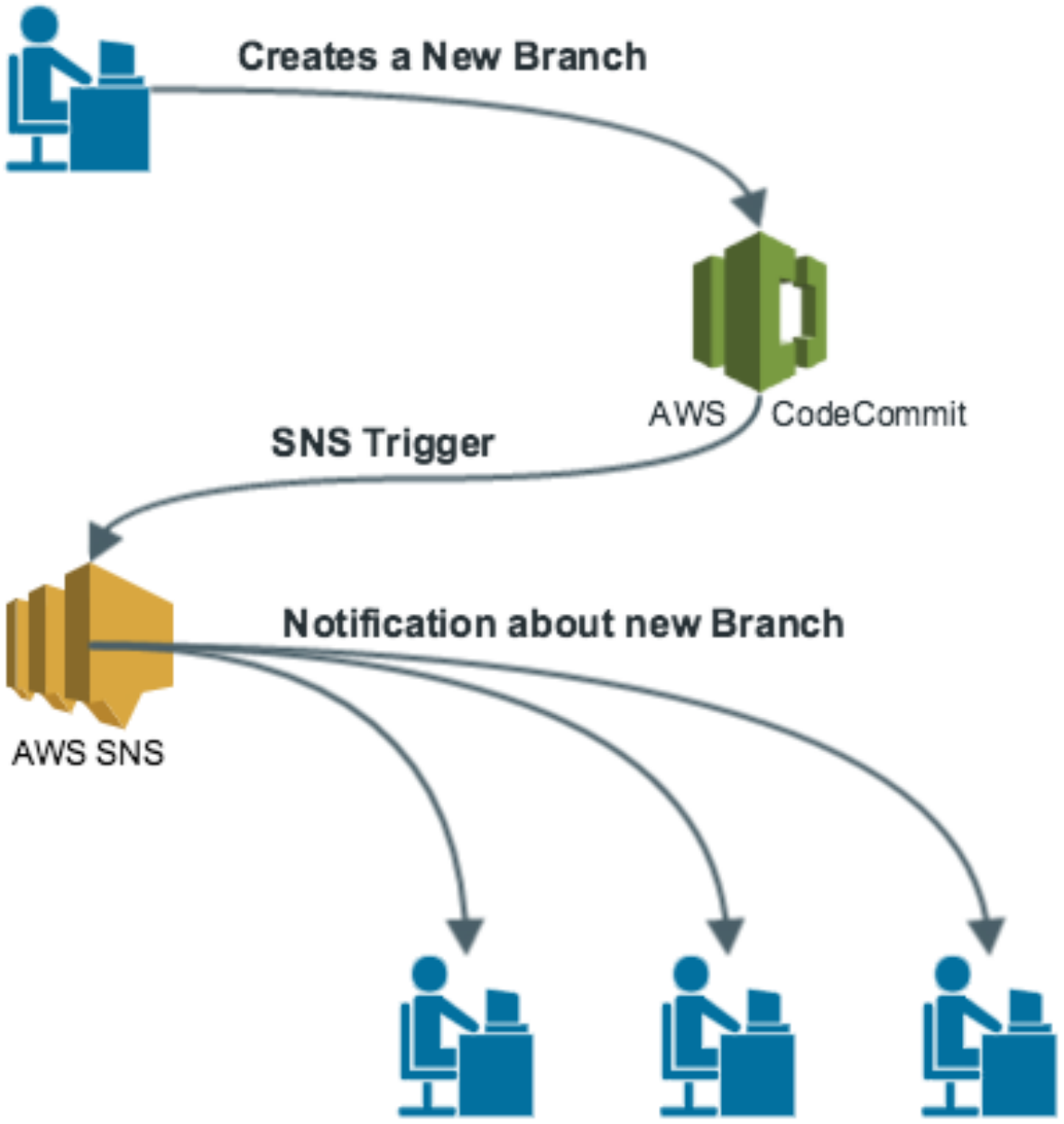
- Triggers are a method where events in CodeCommit can launch automated actions in other AWS services.

- i.e. Creating a new branch can trigger an SNS topic to notify all subscribers that a new branch has been created.

- Currently, you can select one of the following events in CodeCommit to invoke a trigger:
  - *All Repository Events*
  - *A push to existing branch (*master branch)*
  - *The creation of a branch or tag*
  - *The deletion of a branch or tag*

- Triggers can invoke actions in the following AWS services:
  - *AWS Simple Notification Service (SNS)*
  - *AWS Lambda*
  - *You can also take advantage of SNS's ability to integrate with other AWS services like AWS Simple Queue service (SQS)*

What are Triggers? (workflow example)



Linux Academy

Amazon Web Services

CodeCommit & Other AWS Services: Introduction to Triggers

Thank you for watching!

Amazon Web Services

CodeCommit & Other AWS Services: SNS Triggers

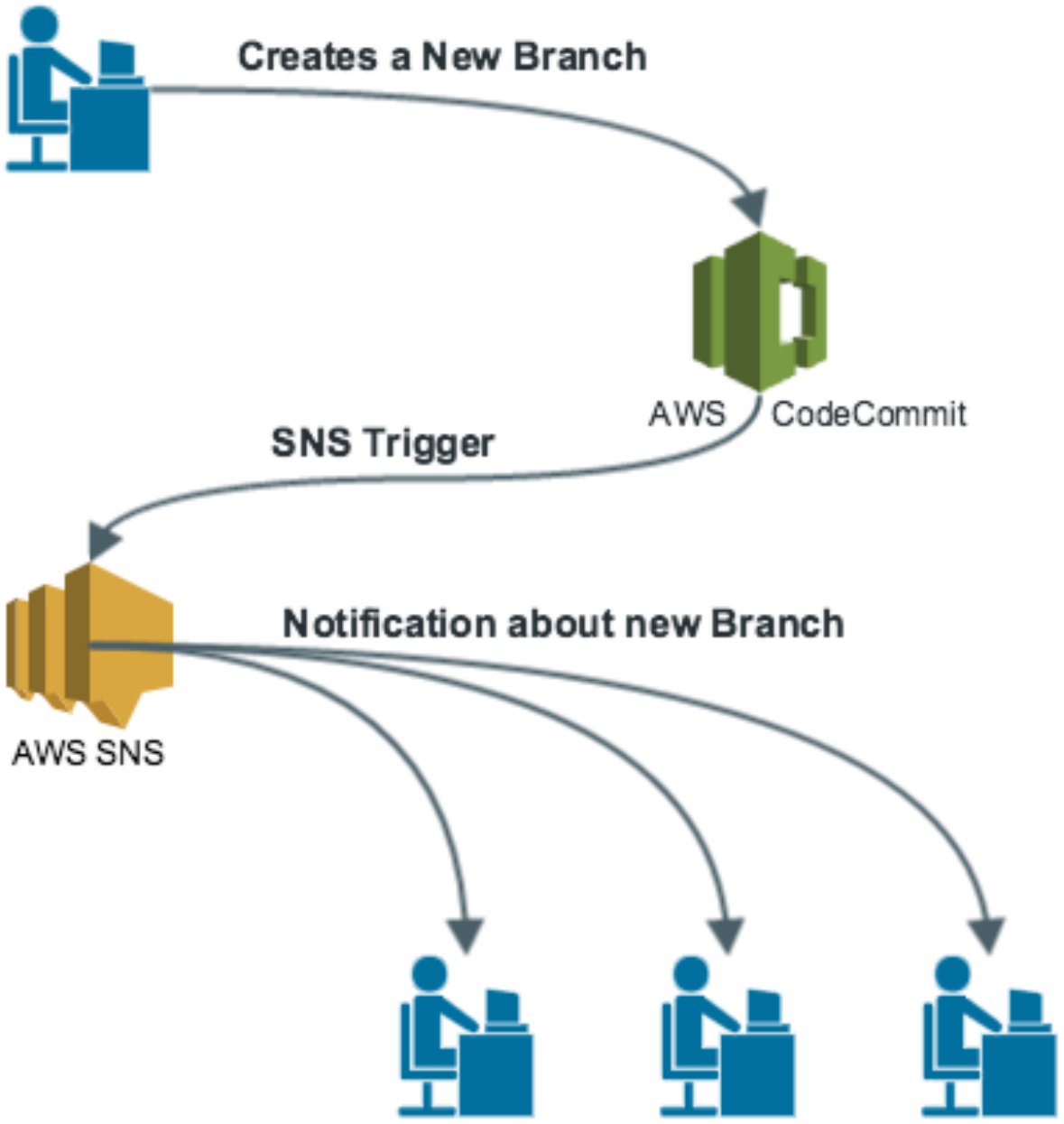
SNS Triggers: Setup/Getting Started

- Triggers can be setup using both:
  - *The AWS console*
  - *The AWS CLI*

- Prerequisites for SNS Triggers:
  - *You must have an existing SNS Tropic*
    - *For in-depth instruction on creating SNS topics, please see the AWS Solution Architect course section on SNS titled "Amazon SNS (Simple Notification Service)" that contains two videos.*
  - *The Repository and the SNS topic must be in the same region (i.e. us-east-1)*

- *Cross-Account Triggers*
  - *If setting up repositories and SNS topics using different root accounts (not different users within the same account), you must have the proper permissions set so CodeCommit can communicate with SNS in the other account.*

Trigger Workflow Example



Create, Push & View an SNS Trigger (CLI)

- Trigger JSON file "Events" list:

all	(all events in the repository)
updateReference	(push occurred)
createReference	(branch or tag created)
deleteReference	(branch or tag deleted)

- For creating, pushing, and viewing triggers – we will use AWS CLI commands NOT Git commands

- Base Related Command:
  aws codecommit

- Based Required Attributes & Syntax:
  Test Trigger before pushing it:
  aws codecommit test-repository-triggers --cli-input-json file://<JSON file>

  Push Trigger to the Repository:
  aws codecommit put-repository-triggers --cli-input-json file://<JSON file>

  View Triggers in the Repository:
  aws codecommit get-repository-triggers --repository-name <Repo Name>

Linux Academy

Amazon Web Services

CodeCommit & Other AWS Services: SNS Triggers

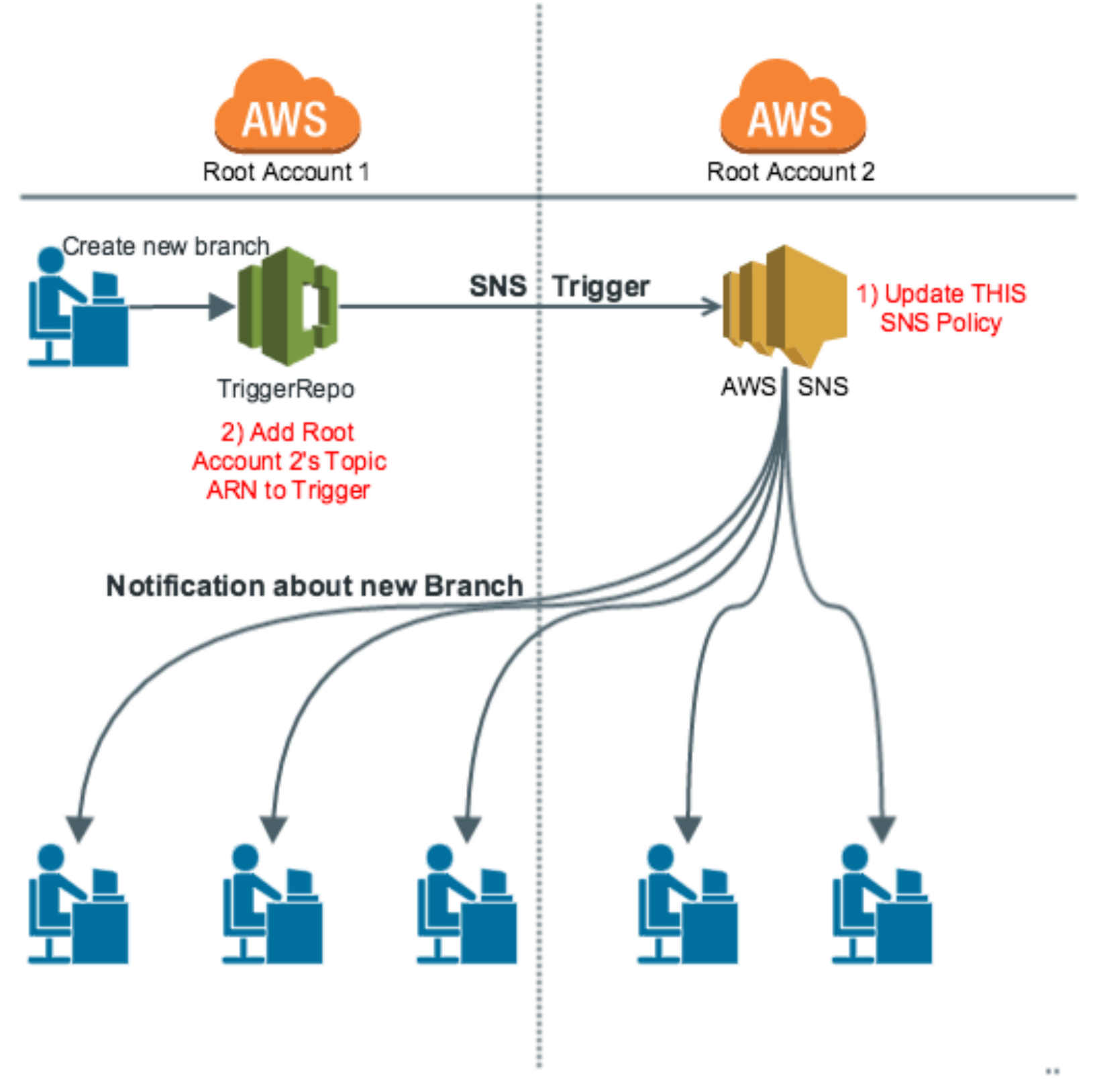
Thank you for watching!

Amazon Web Services

CodeCommit & Other AWS Services: Cross-Account SNS Triggers

Linux Academy

Cross-Account SNS Trigger



Amazon Web Services

CodeCommit & Other AWS Services: Cross-Account SNS Triggers

Thank you for watching!

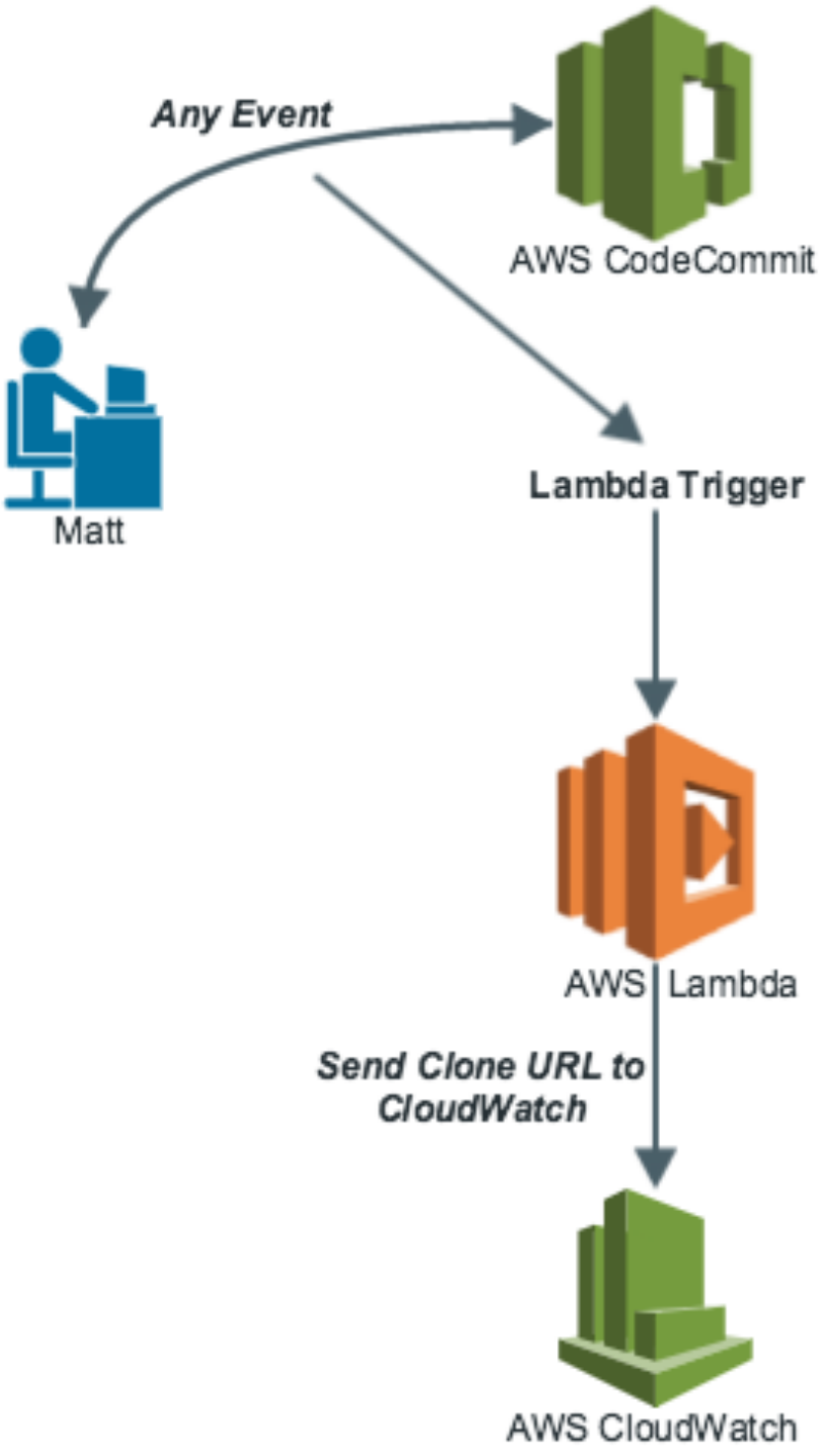
Amazon Web Services

CodeCommit & Other AWS Services: Lambda Triggers

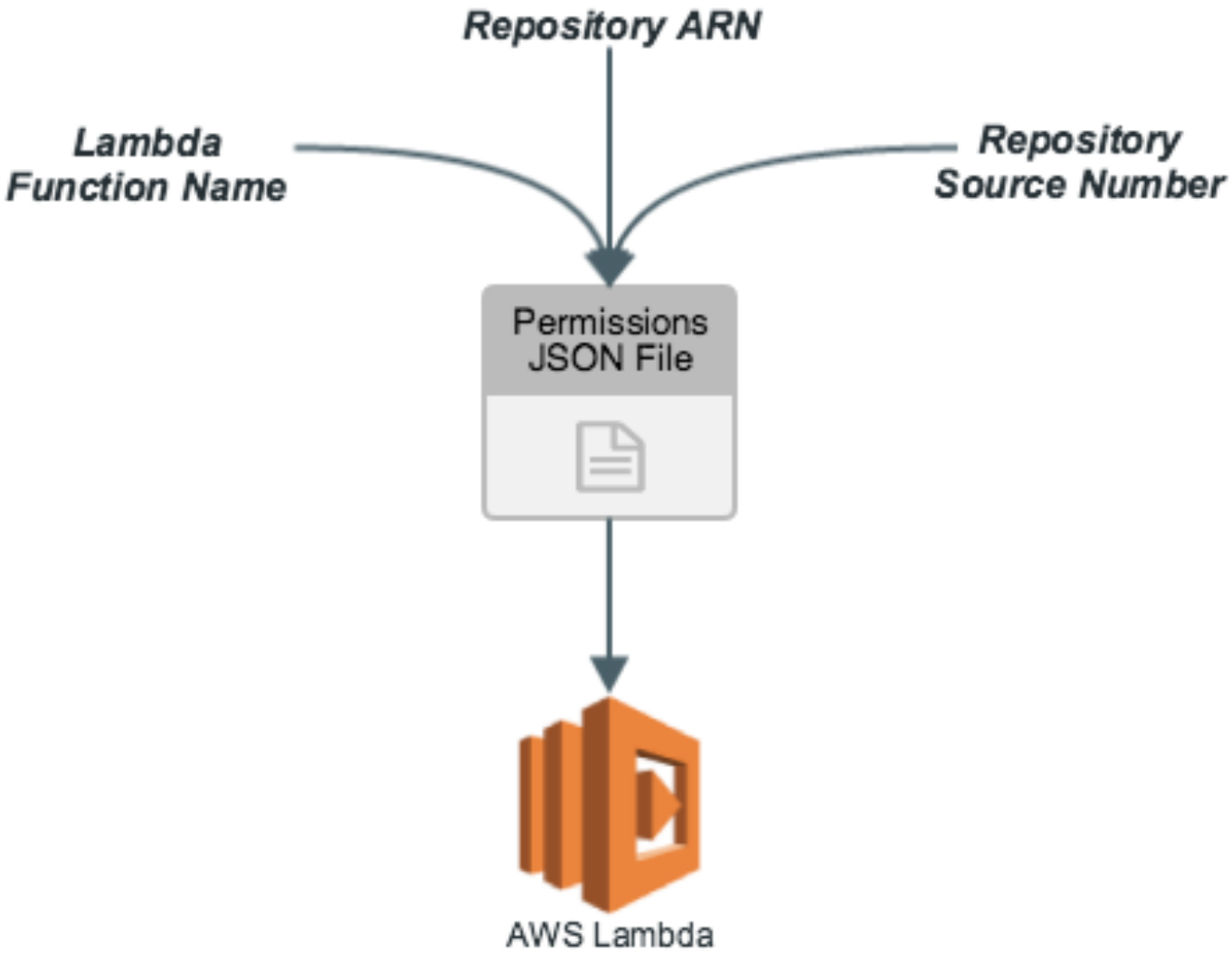
Lambda Triggers: Getting Started

- What is AWS Lambda
  - *AWS Lambda is a relatively new feature to AWS and is a compute service where you can upload your code to Lambda and the service can run the code on your behalf using AWS infrastructure.*
  - *For further details on AWS Lambda, see our full AWS Lambda course titled "Lambda Deep Dive".*
- Triggers can be setup using both:
  - *The AWS console*
  - *The AWS CLI*
- Prerequisites for Lambda Triggers:
  - *You must have an existing Lambda function created*
  - *The repository and the Lambda function must be in the same region (i.e. us-east-1)*

Lambda Trigger Workflow (example)



Setting up Lambda Permissions



aws lambda add-permission --cli-input-json file://<My Permissions JSON File>

Create, Push & View an Lambda Trigger (CLI)

- Trigger JSON file "Events" list:

all	(all events in the repository)
updateReference	(push occurred)
createReference	(branch or tag created)
deleteReference	(branch or tag deleted)

- For creating, pushing, and viewing triggers – we will use AWS CLI commands NOT Git commands

- Base Related Command:
  aws codecommit

- Based Required Attributes & Syntax:
  Test Trigger before pushing it:
  aws codecommit test-repository-triggers --cli-input-json file://<JSON file>

  Push Trigger to the Repository:
  aws codecommit put-repository-triggers --cli-input-json file://<JSON file>

  View Triggers in the Repository:
  aws codecommit get-repository-triggers --repository-name <Repo Name>

Linux Academy

Amazon Web Services

CodeCommit & Other AWS Services: SNS Triggers

Thank you for watching!


Linux Academy
Amazon Web Services
CodeCommit & Other AWS Services:
AWS Key Management System (KMS)

What is AWS Key Management System (KMS)?

- Amazon's encryption management service that allows you to easily create and control the keys used to encrypt your data.

- You can:
  - Create, describe, enable and disable master keys
  - Set master key usage polices and logs
  - Encrypt and decrypt your data using the master keys you create

CodeCommit & KMS

- Key creation:
  - AWS automatically creates an AWS-managed key when you create a CodeCommit repository in a given region.
  - The key is stored in the same region as the repository.
  - The key is stored in your AWS account.

- Key usage:
  - The key (created specifically for CodeCommit), is then used to encrypt and decrypt your data while both In-Transit, and when at rest.
  - In-Transit:
    - AWS encrypts the data when you push to a repository (***git push***)
    - AWS decrypts the data when you pull from a repository (***git pull***)

- Encryption Type:
  - AWS KMS uses AES-256

NOTE: CodeCommit integrates with AWS KMS automatically. You do not have to take any steps yourself, to encrypt your data when using CodeCommit (while in-transit or at rest on AWS servers)

CodeCommit, KMS & IAM Policies

- CodeCommit performs these actions against the default key (aws/codecommit)
  - "kms:Encrypt"
  - "kms:Decrypt"
  - "kms:ReEncrypt"
  - "kms:GenerateDataKey"
  - "kms:GenerateDataKeyWithoutPlaintext"
  - "kms:DescribeKey”

- The IAM user used to create a repository (and thus the key for CodeCommit) does not need to be given permission to perform these actions, but you also should be mindful not to attach any polices that denies any of these actions.


Linux Academy
Amazon Web Services
CodeCommit & Other AWS Services:
AWS Key Management System (KMS)
Thank you for watching!