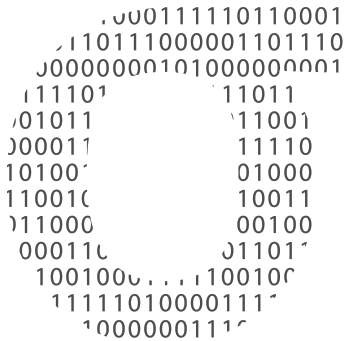


SIGMADATA
Learning

Azure DevOps for BI

Introduction

SIGMADATA Learning

- Please introduce yourself:
  - Name
  - Company
  - Why you attend this course
  - What you expect to learn

Course Overview (day 1)

SIGMADATA
Learning

- Introduction to CI / CD
- Overview of Azure DevOps
- Using Git & Azure Repos: Basics
- Using Git: developing on branches

Course Overview (day 2)

SIGMADATA Learning

- *(Using Azure Pipelines)*
- Setting up projects
- Setting up build & release pipelines
- Building and deploying BI projects

Environment

SIGMADATA Learning

VM
= devmachine

SQL Database DEV
- DW
- SSISDB

SQL Database
- DW
- SSISDB

Azure Data Factory DEV
- Orchestration
- SSIS-IR for ETL

Azure Data Factory
- Orchestration
- SSIS-IR for ETL

Storage account
- Landing zone
- staging

SIGMADATA
Learning

Introducing CI / CD

Azure DevOps for BI & Data Platforms

Module 1

Introduction to CI / CD

SIGMADATA
Learning

- CI = Continuous Integration
  - Why Continuous?
- What is meant by “CD”?
  - Continuous Delivery?
  - Continuous Deployment?
  - Continuous …?

What is CI again?

SIGMADATA
Learning

- CI = Continuous Integration
  - Build
- CD = Continuous Delivery
  - Release
- “Traditional” software development vs BI

SIGMADATA
Learning

Continuous Integration

Integrating all developed code in one shared repo.

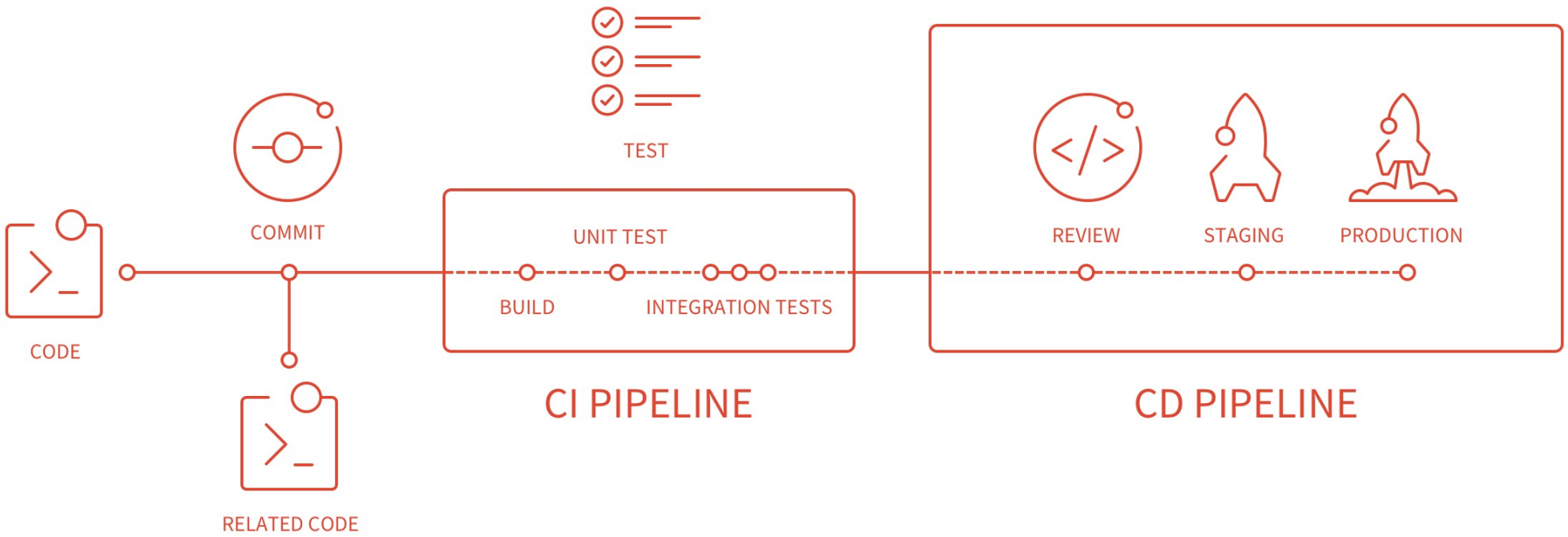
Build it to create artifacts.

SIGMADATA
Learning

Continuous Delivery

Ensure software can reliably be released.
At any time.

CI / CD



Source: https://docs.gitlab.com/ee/ci/

Overview of build / release steps (traditional)

SIGMADATA Learning

Build (CI)

- Build (= compile + link)
- Automated unit tests
- Automated integration tests

Release (CD)

	D	T	A	P
Deploy artifacts	X	X	X	X
Manual test scripts (developer)		X	X	
Manual test scripts (user)			X	

Release can be dependent on:

- Test results
- User approval

Are Data Engineering projects different?

SIGMADATA Learning

- Who uses automated unit testing?
  - … for all your code and processes
- Can we do integration testing at build time?
  - ... without deploying the artifacts
- What is a “unit for testing”?
  - Click a button? Call a method
  - Translating 1B rows in an ETL process..?
- Database unit tests are available, but…
  - … mainly focused at transactional systems

Translate CI / CD to Data Engineering projects

SIGMADATA Learning

Build (CI)

- Build (= compile / link)
- Automated unit tests
- Automated integration tests

Release (CD)

	D	T	A	P
Deploy artifacts	X	X	X	X
Manual test scripts (developer)		X	X	
Manual test scripts (user)			X	

Translate CI / CD to Data Engineering projects

SIGMADATA Learning

Build (CI)

- Build (= compile + link)
- **Create artifacts**
- Automated unit tests
- Automated integration tests

Release (CD)

	D	T	A	P
Deploy artifacts	X	X	X	X
Validate ETL		X		
Manual test scripts (developer)		X	X	
Manual test scripts (user)			X	

Translate CI / CD to Data Engineering projects

SIGMADATA Learning

Build (CI)

- ~~Build (= compile + link)~~
- **Create artifacts**
- Automated unit tests
- Automated integration tests

Release (CD)

	D	T	A	P
Deploy artifacts	X	X	X	X
Validate ETL		X		
Manual test scripts (developer)		X	X	
Manual test scripts (user)			X	

Translate CI / CD to Data Engineering projects

SIGMADATA Learning

Build (CI)

- ~~Build (= compile + link)~~
- Create artifacts
- ~~Automated unit tests~~
- Automated integration tests

Release (CD)

	D	T	A	P
Deploy artifacts	X	X	X	X
Validate ETL		X		
Manual test scripts (developer)		X	X	
Manual test scripts (user)			X	

Translate CI / CD to Data Engineering projects

SIGMADATA Learning

Build (CI)

- ~~Build (= compile + link)~~
- Create artifacts
- ~~Automated unit tests~~
- Automated integration tests

Release (CD)

	D	T	A	P
Deploy artifacts	X	X	X	X
Validate ETL		X		
Automated integration tests		X		
Manual test scripts (developer)		X	X	
Manual test scripts (user)			X	

Release can be dependent on:
- Test results
- User approval

Translate CI / CD to Data Engineering projects

SIGMADATA Learning

Build (CI)

- ~~Build (= compile + link)~~
- Build (create artifacts)
- ~~Automated unit tests~~
- ~~Automated integration tests~~

Release (CD)

	D	T	A	P
Deploy artifacts	X	X	X	X
Validate ETL		X		
Automated integration tests		X		
Automated regression tests			X	
Manual test scripts (developer)		X	X	
Manual test scripts (user)			X	

Release can be dependent on:
- Test results
- User approval

Translate CI / CD to Data Engineering projects

SIGMADATA Learning

Build (CI)

- Build (create artifacts)

Release (CD)

	D	T	A	P
Deploy artifacts	X	X	X	X
Validate ETL		X		
Automated integration tests		X		
Automated regression tests			X	
Manual test scripts (developer)		X	X	
Manual test scripts (user)			X	

Release can be dependent on:

- Test results
- User approval

How to achieve CI / CD for Data Engineering projects?

SIGMADATA Learning

Continuous Integration

- Create deployable artifacts.
- Integrate database solutions (check integrity using dacpac)

Continuous Delivery

- Deploy to DTAP environment in a pipeline
- Validate ETL logic
- Integration tests after deployment to specific environments (NBi etc.)

SIGMADATA
Learning

Overview of Azure DevOps

Azure DevOps for BI & Data Platforms

Overview of Azure DevOps

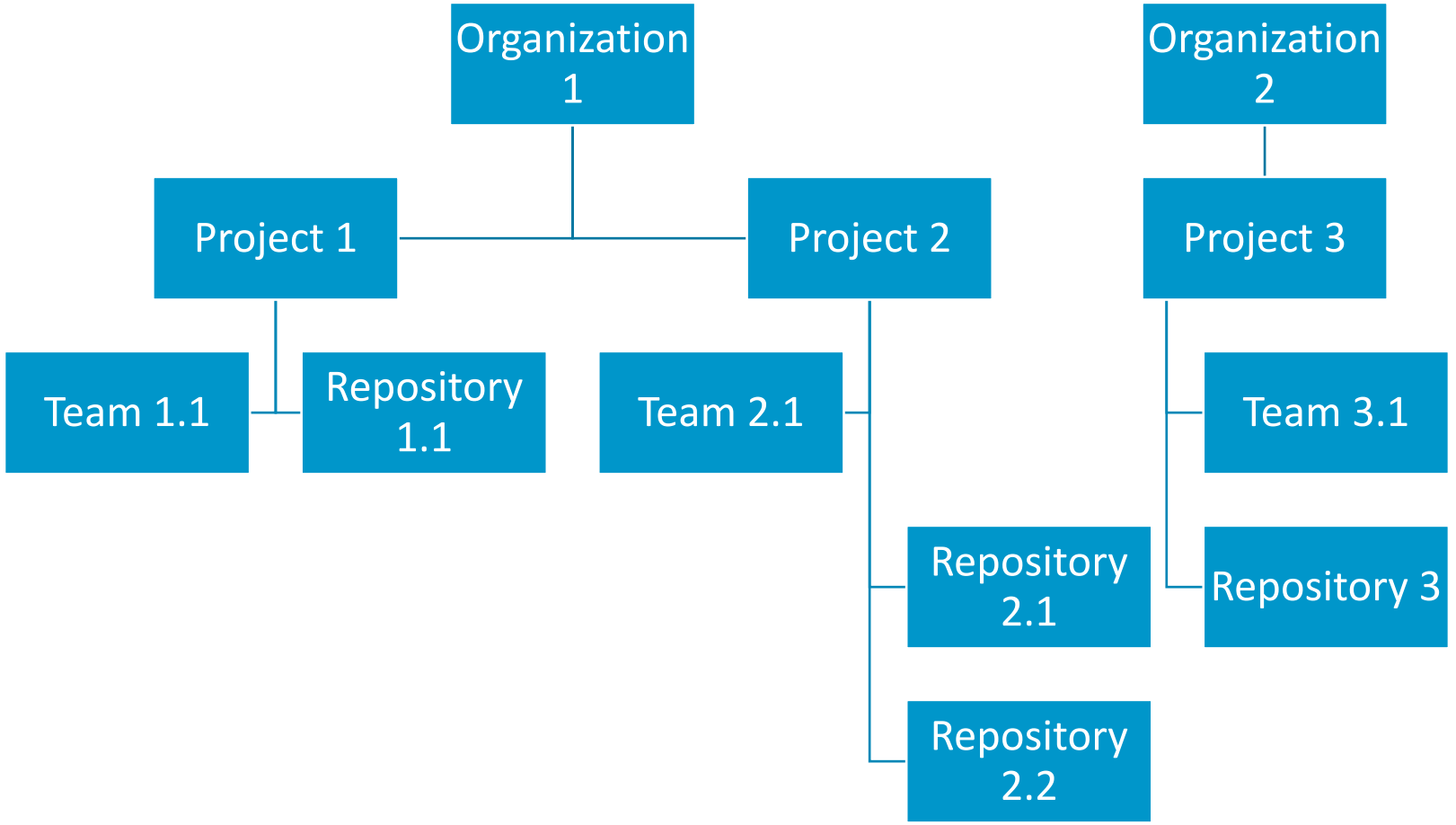
SIGMADATA Learning

https://azure.microsoft.com/en-us/services/devops/

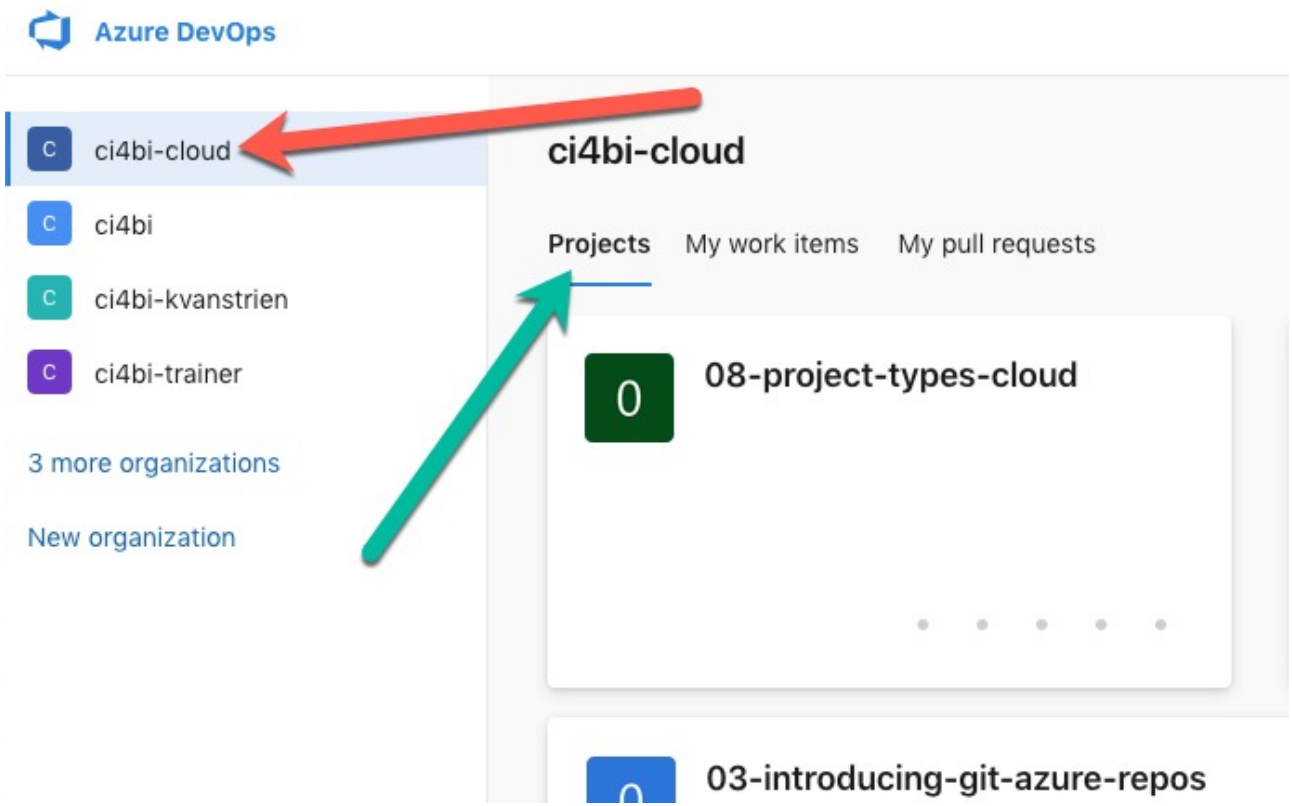
- Agile tools to support planning & track work (Kanban, Scrum)
- Build & release management
- Git repositories for source control
- Tools for testing
  - Manual testing
  - Load testing
  - Continuous testing
- Artifacts management / package repositories
- Extensions (integrates with Campfire, Slack, Trello, Uservoice, etc.)

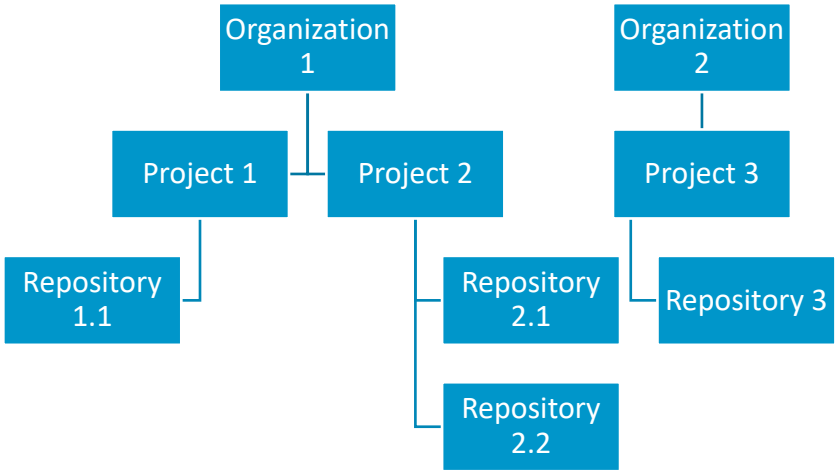
Organizations, projects and repositories

SIGMADATA Learning



Organizations, projects and repositories





Organizations

SIGMADATA Learning

- Mechanism for organizing / connecting groups of related projects
  - Per company? Per business unit? For you?
  - Best to start with one
  - Each organization has its own ”free tier”
    - 1800 minutes hosted Pipeline job / month + one self-hosted job
    - Boards
    - Unlimited private repos
    - Artifacts
    - Load testing
    - Unlimited stakeholders
    - (up to 5 “Basic” users per service type)

https://docs.microsoft.com/en-us/azure/devops/user-guide/plan-your-azure-devops-org-structure?view=azure-devops

Projects

SIGMADATA Learning

- Container for
  - Boards and backlogs
  - Pipelines for CI/CD
  - Repos
  - Continuous Test integration
- Single vs. multiple?
  - Single with many repos / teams
  - Many with own sets of repos, builds, workitems, etc.?
- For granular security, choose multiple
- Multiple teams → multiple boards

https://docs.microsoft.com/en-us/azure/devops/organizations/projects/about-projects?view=azure-devops

Repositories

SIGMADATA Learning

- Exist within Project
- Git or TFVC
- Git
  - No limit on number of Git repos
  - Decentralized (more on that later)
  - “One repo per independently deploy-able product or service”
- TFVC
  - One big repository
- One vs. Many

SIGMADATA
Learning

Overview of Azure DevOps

Interface tour

Azure DevOps Interface Tour

SIGMADATA Learning

- Organizations, Projects, Repositories
- Boards
  - Board view
  - Backlog view
- Pipelines
  - Build pipeline
  - Release pipeline
- Test Plans
- Artifacts?
- Overview: Summary, Dashboards, Wiki

Overview of Azure DevOps - Summary

SIGMADATA Learning

- Most important components:
  - Repos – contains repositories (VS Solutions, projects, etc.)
  - Boards – contains planning & work
  - Pipelines – build & deployment automation
- When checking in code, refer to a work item (or task/feature/etc.)

SIGMADATA
Learning

Introducing Git & Azure Repos

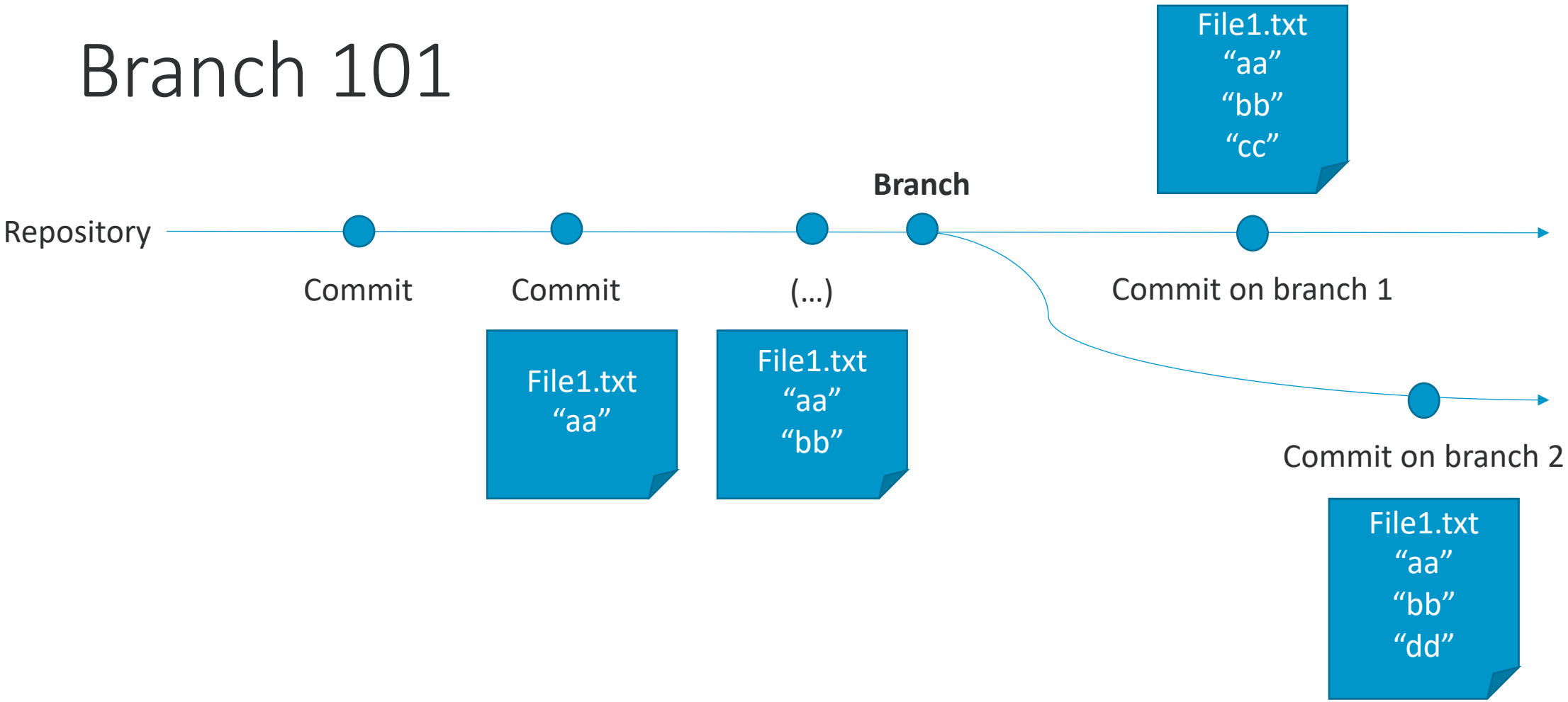
Azure DevOps for BI & Data Platforms

About Git

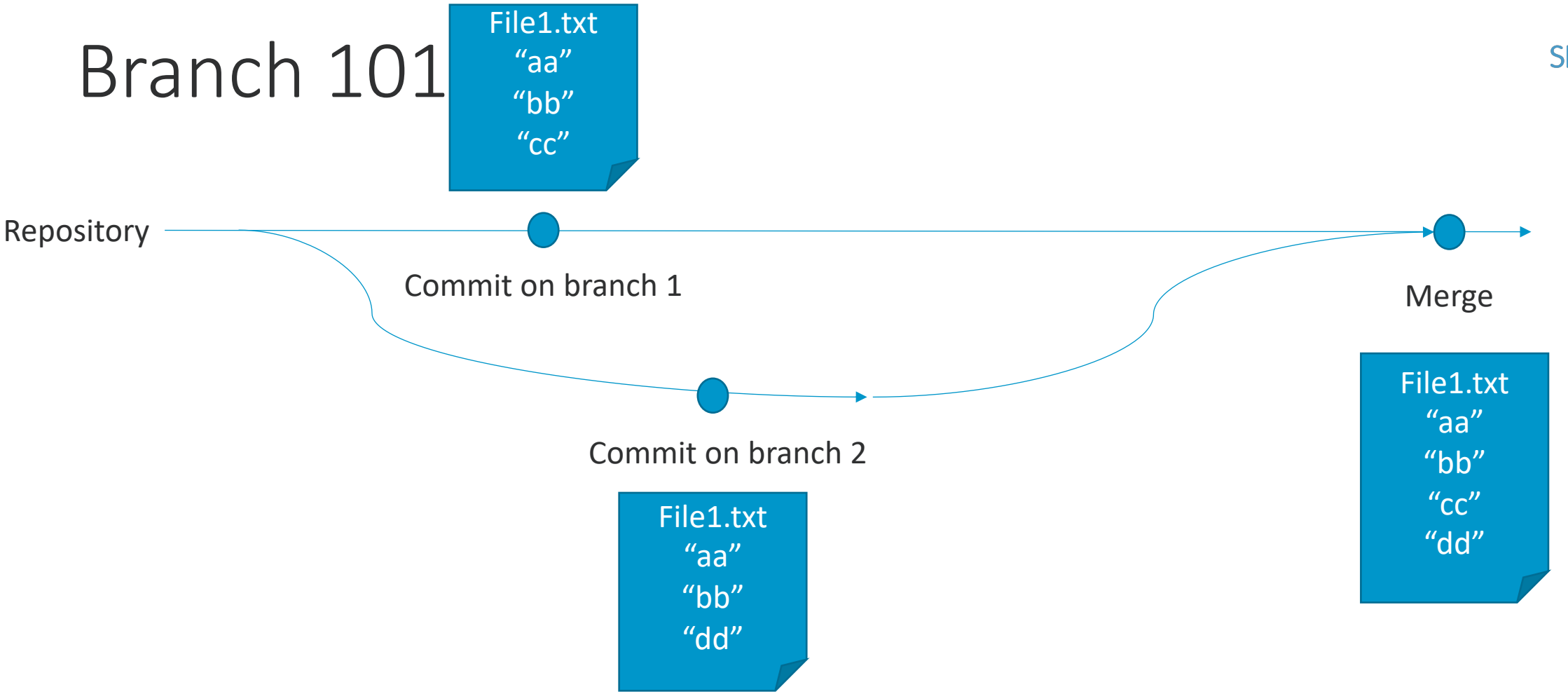
SIGMADATA
Learning

- Version Control System (VCS)
- Differences between Git and other VCSes
  - Distributed (decentralized)
  - Built for branching & merging
  - Allow modification of previous commits (rebase)

Branch 101


File1.txt
“aa”
“bb”
“cc”
Branch
Repository
Commit
Commit
(…)
Commit on branch 1
File1.txt
“aa”
File1.txt
“aa”
“bb”
Commit on branch 2
File1.txt
“aa”
“bb”
“dd”

Branch 101


File1.txt
“aa”
“bb”
“cc”
Repository
Commit on branch 1
Merge
Commit on branch 2
File1.txt
“aa”
“bb”
“dd”
File1.txt
“aa”
“bb”
“cc”
“dd”

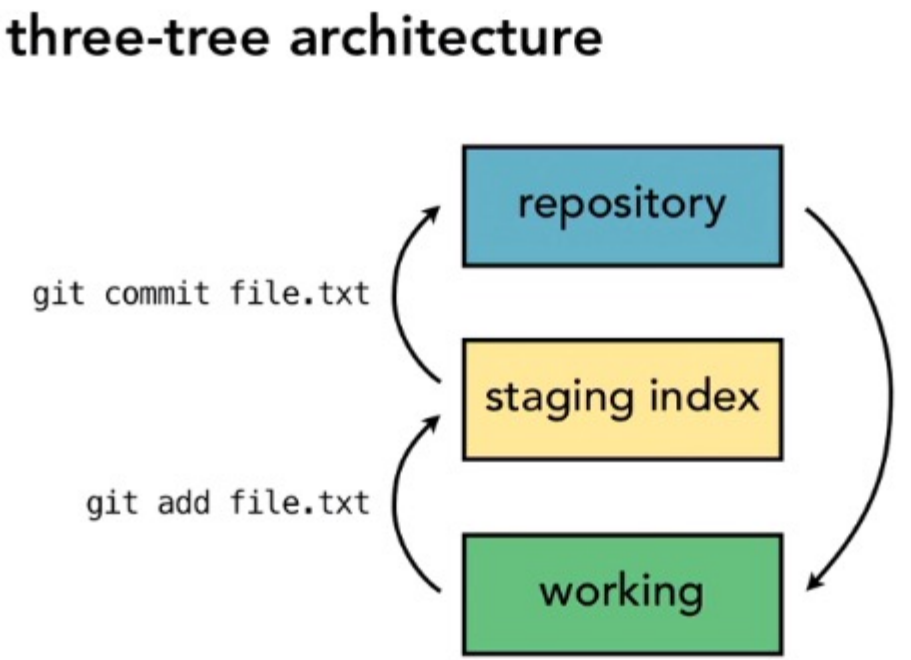
Git 101 – participation demo

SIGMADATA Learning

- Trainer creates git repo
  - Shows how to add a file
- Trainees “clone” this repo
  - Where’s the file?
- Trainer explains commit
- Trainees pull file
- Trainer explains concept of “three tier architecture”

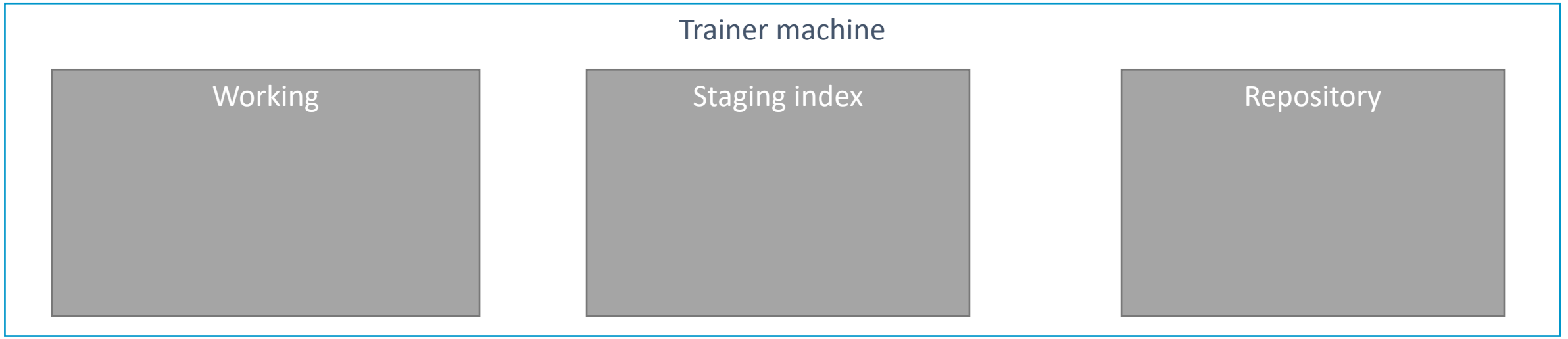
Three main sections / states for your files

1. Modified: Working directory
2. Staged: Staging directory
3. Committed: Repository

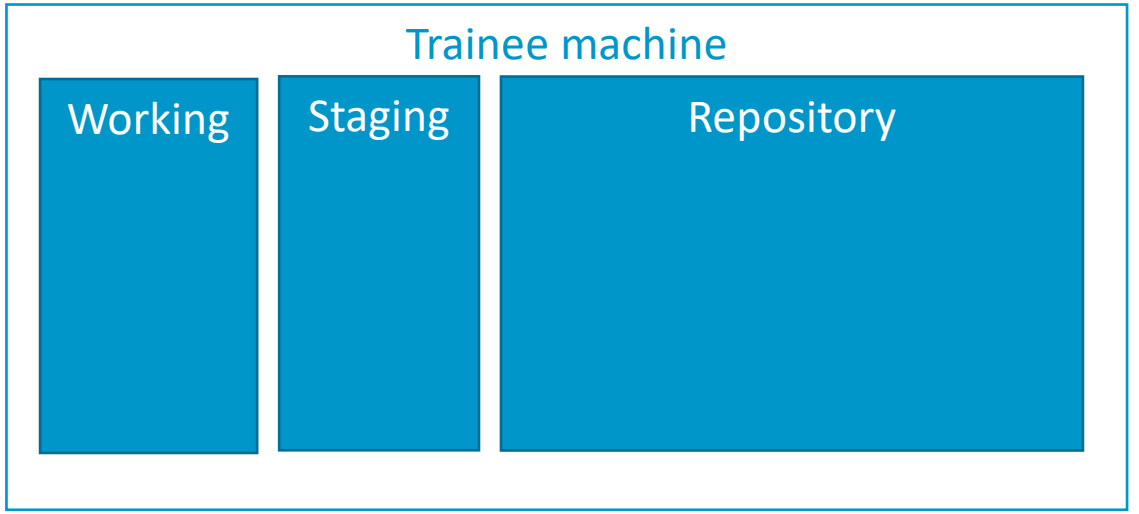


https://www.cloudways.com/blog/git-tutorial-for-beginners-version-control/

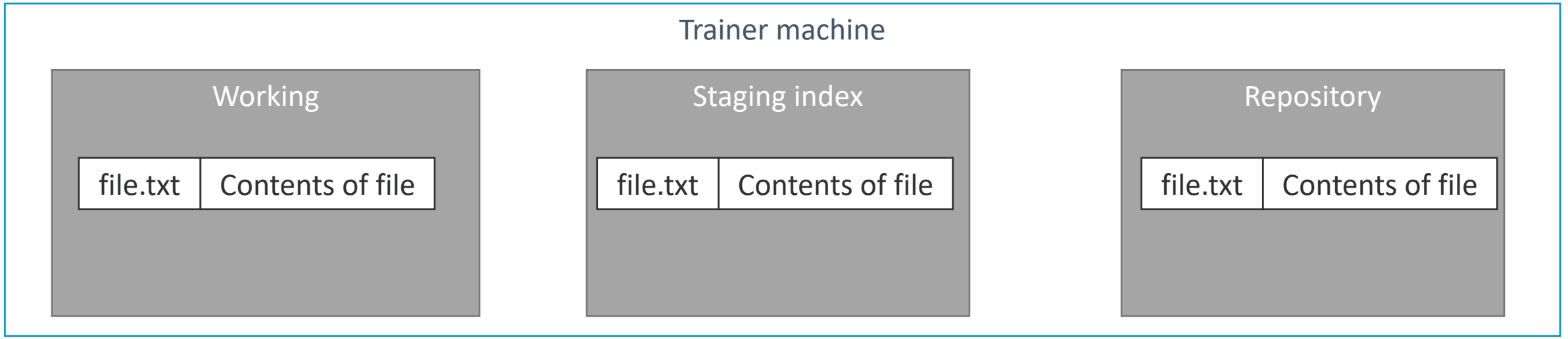
Git 101



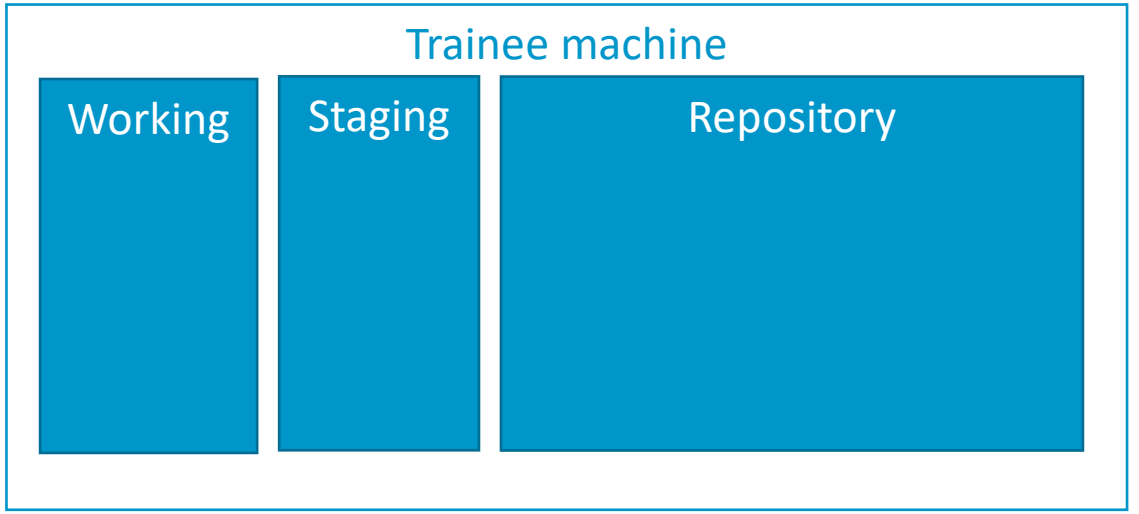
1. git init
2. git clone path/to/other/repo



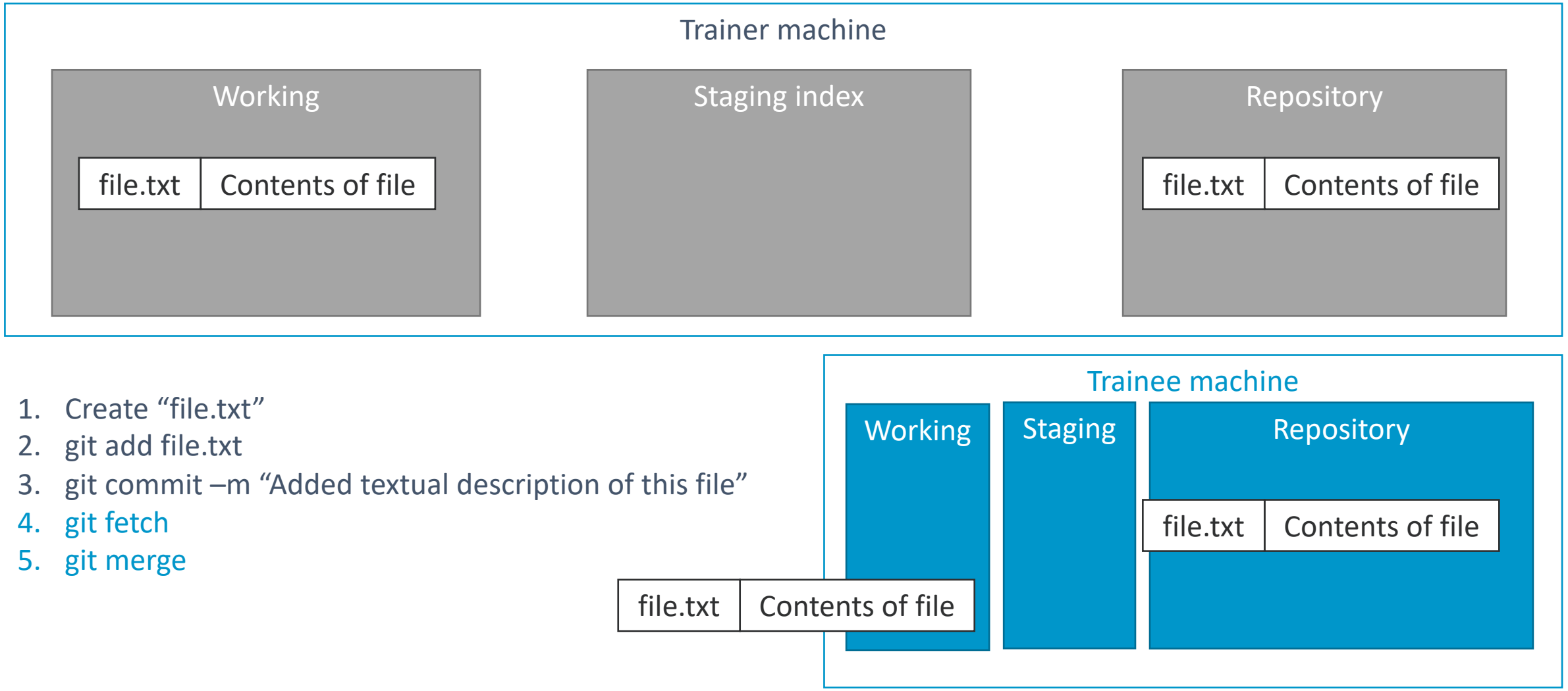
Git 101



1. *Create "file.txt"*
2. git add file.txt
3. git commit –m "Added textual description of this file"



Git 101



1. Create “file.txt”
2. git add file.txt
3. git commit –m “Added textual description of this file”
4. git fetch
5. git merge

Working with Git

SIGMADATA Learning

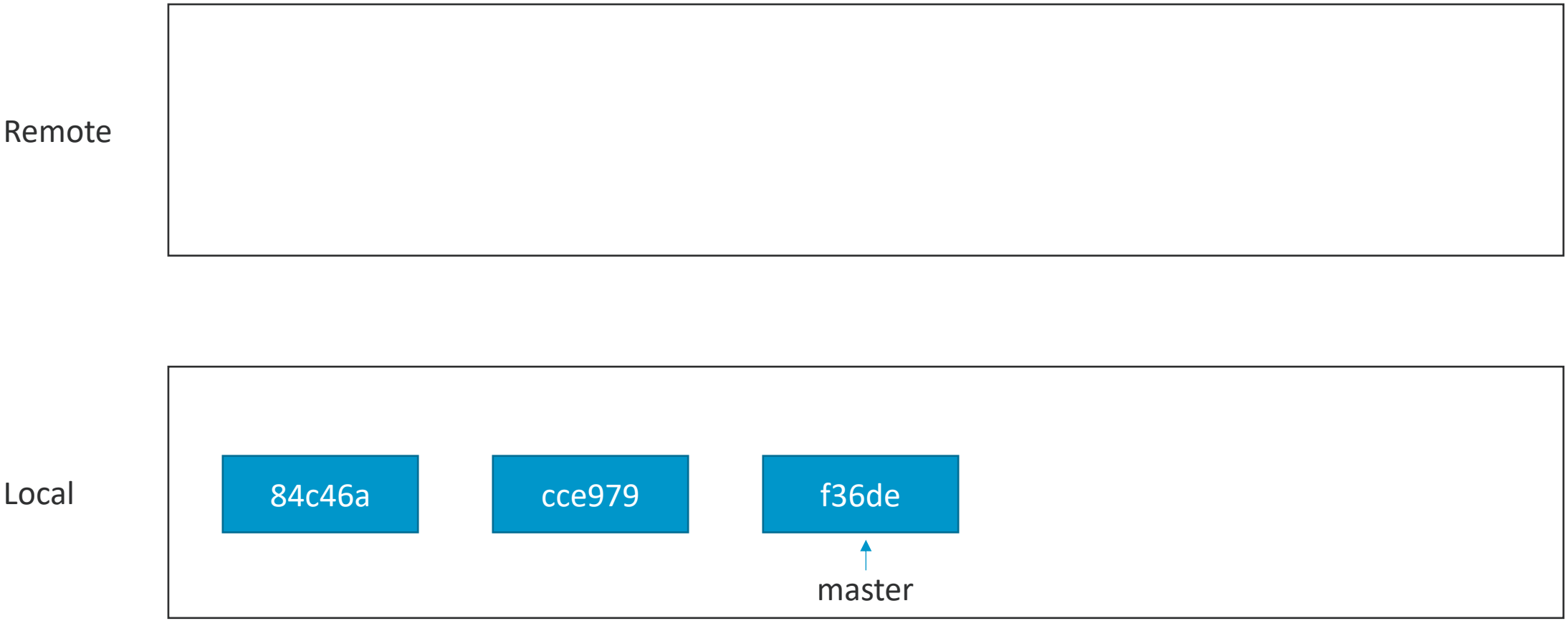
- Trainees add files, commit and push
- Trainer explains concept of “blessed repo”
- Trainees clone this “blessed repo”
  - Trainer explains this is the way Azure DevOps works: it contains the “blessed repo”
- Trainees modify files, try to push -> rejected. First, pull!
- Trainer explains push, pull, fetch and merge

Git so far

SIGMADATA Learning

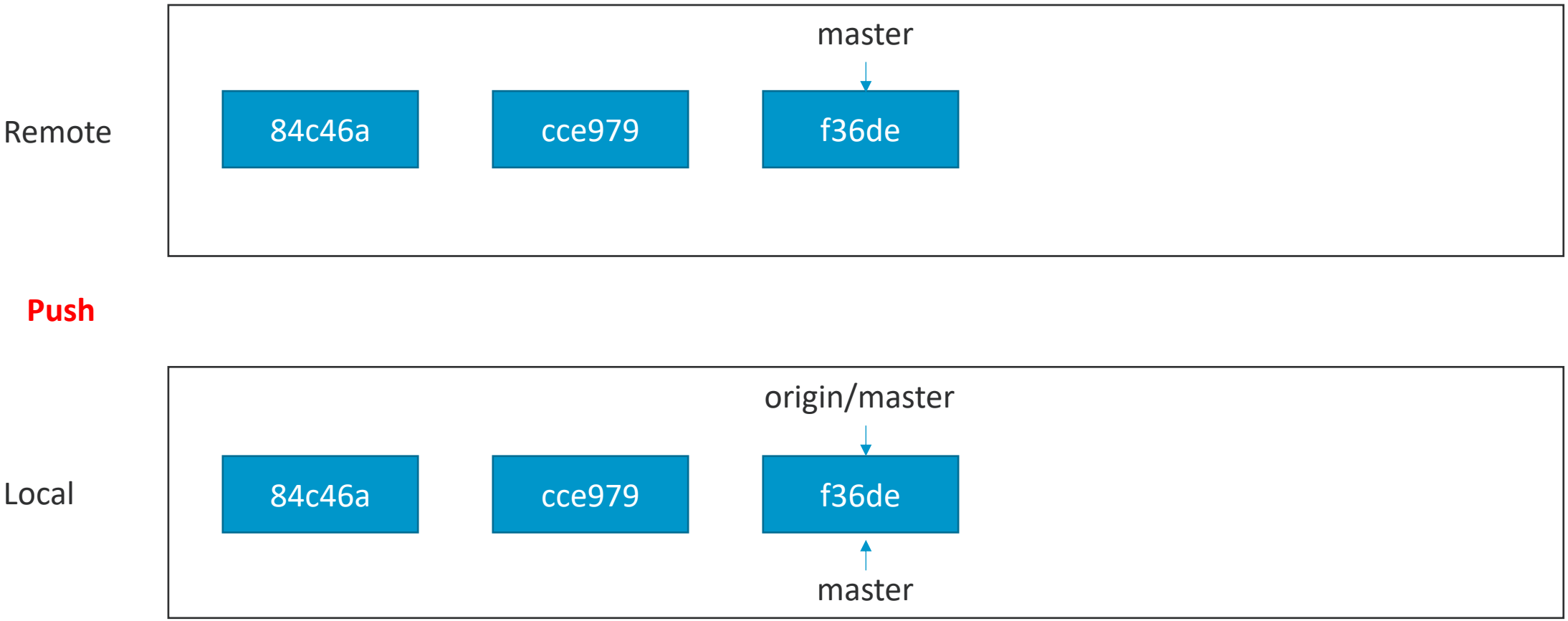
- (Should be) version control
- **Create** a Git repository:
  - Empty with **git init**
  - Clone an existing with **git clone**
- **Add files** to staging index:
  - **git add** (multiple files can be added at once)
- **Commit** staging:
  - **git commit** (editor will pop up)
  - **git commit –m “Commit Message”** (no editor needed)
- **View commits**:
  - **git log**
- **git status**

Using local & remote repositories



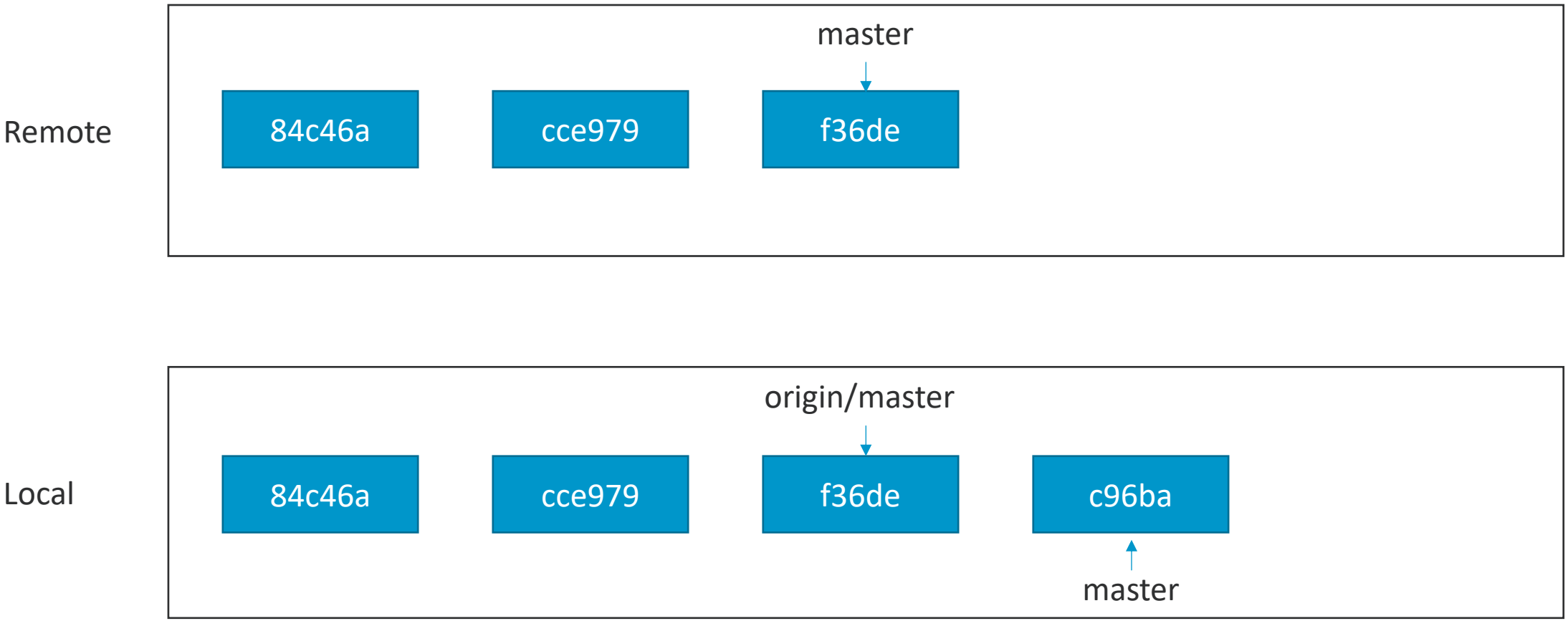
Designed after sample GIF by Kevin Skoglund:
https://www.lynda.com/Git-tutorials/Using-local-remote-repositories/100222/111327-4.html

Using local & remote repositories



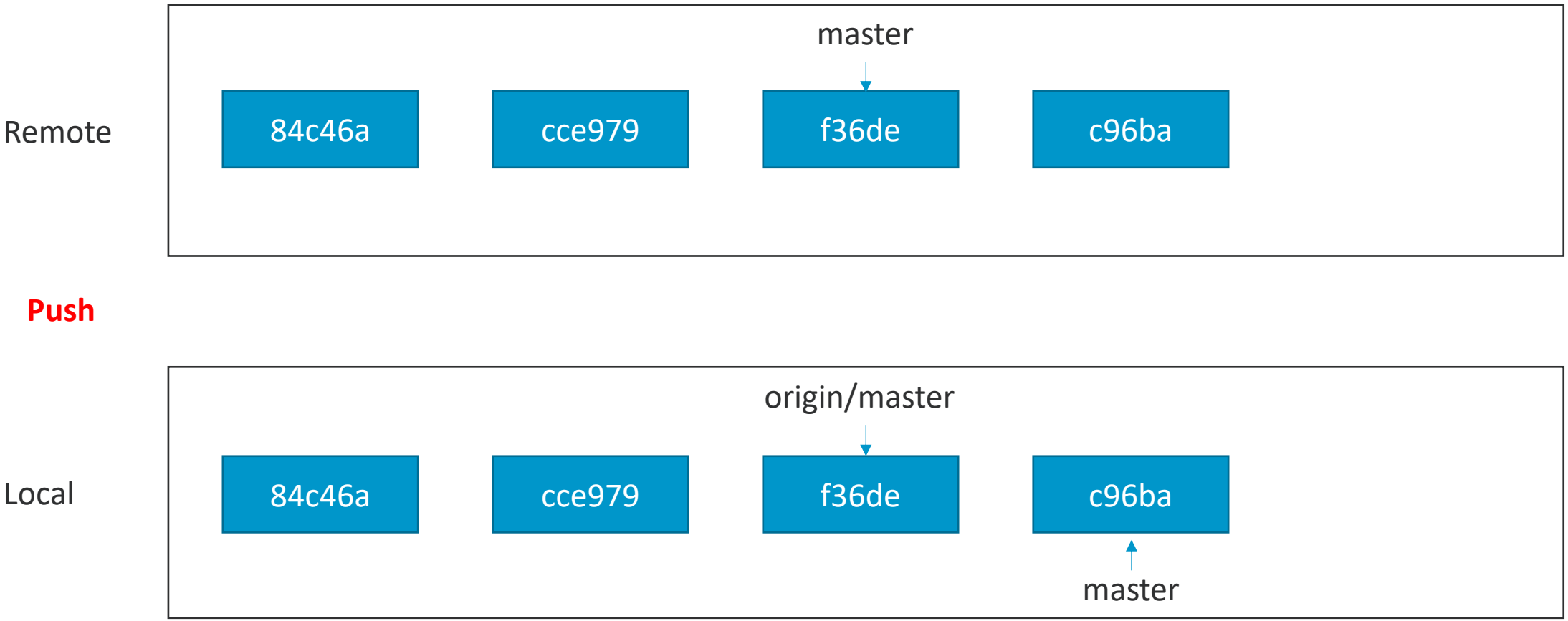
Designed after sample GIF by Kevin Skoglund:
https://www.lynda.com/Git-tutorials/Using-local-remote-repositories/100222/111327-4.html

Using local & remote repositories



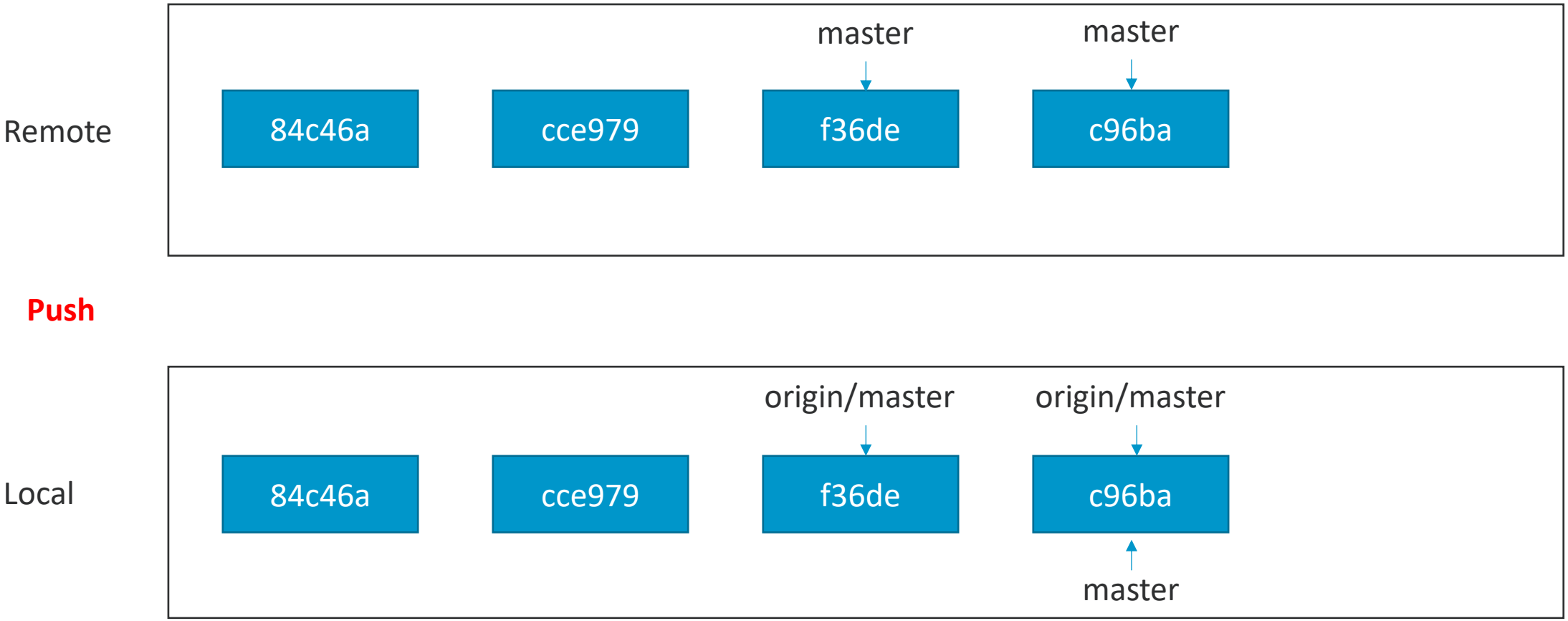
Designed after sample GIF by Kevin Skoglund:
https://www.lynda.com/Git-tutorials/Using-local-remote-repositories/100222/111327-4.html

Using local & remote repositories



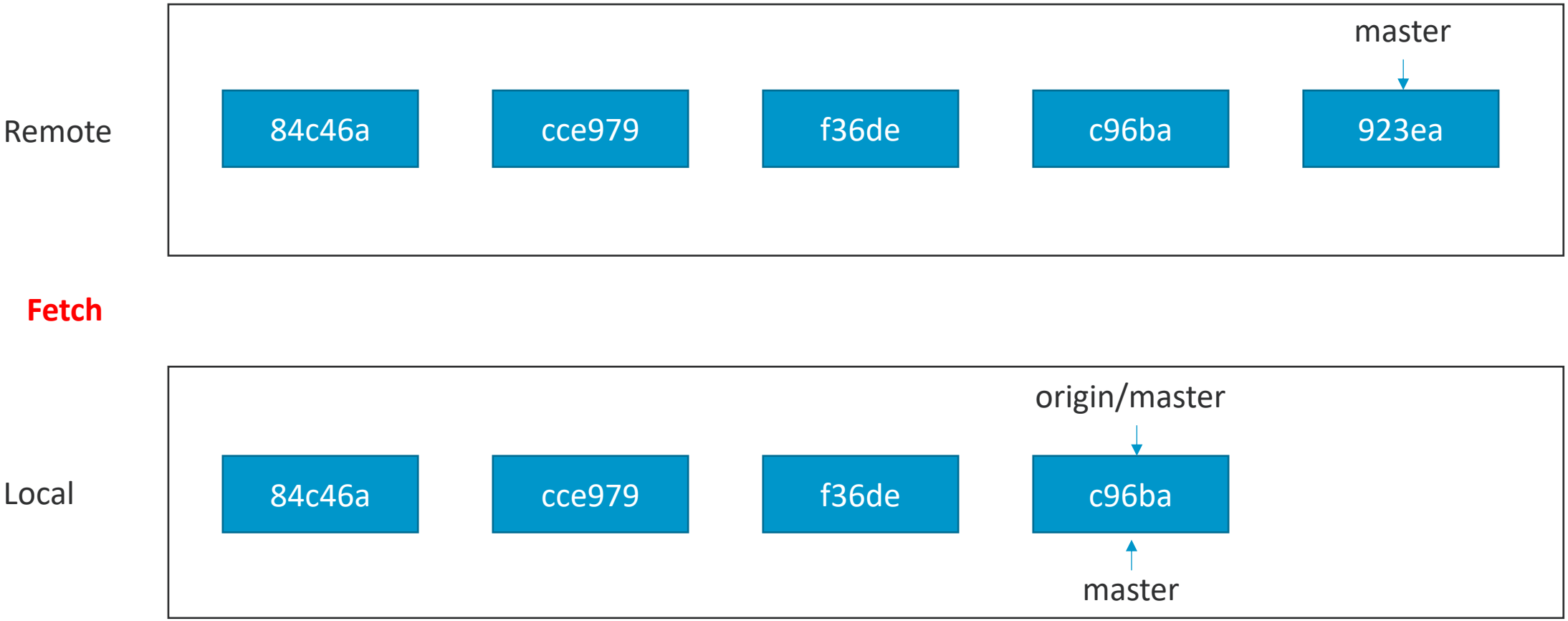
Designed after sample GIF by Kevin Skoglund:
https://www.lynda.com/Git-tutorials/Using-local-remote-repositories/100222/111327-4.html

Using local & remote repositories



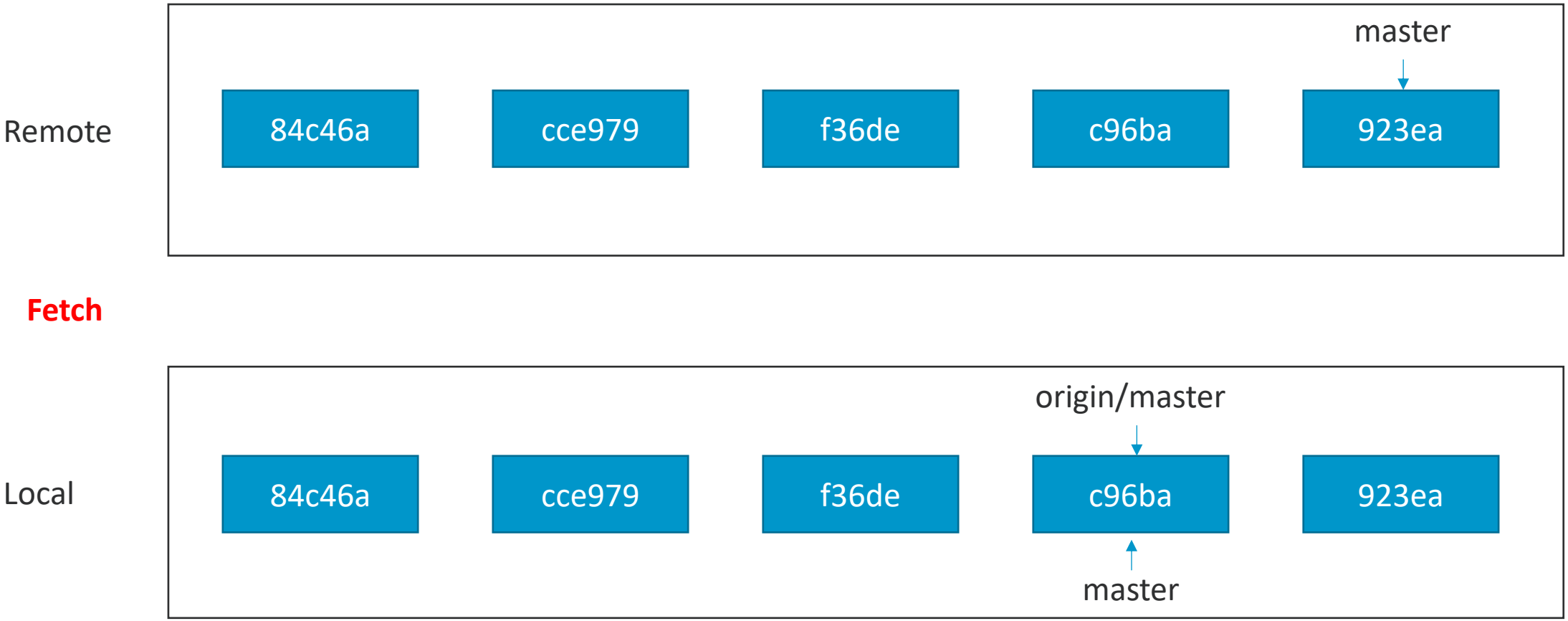
Designed after sample GIF by Kevin Skoglund:
https://www.lynda.com/Git-tutorials/Using-local-remote-repositories/100222/111327-4.html

Using local & remote repositories



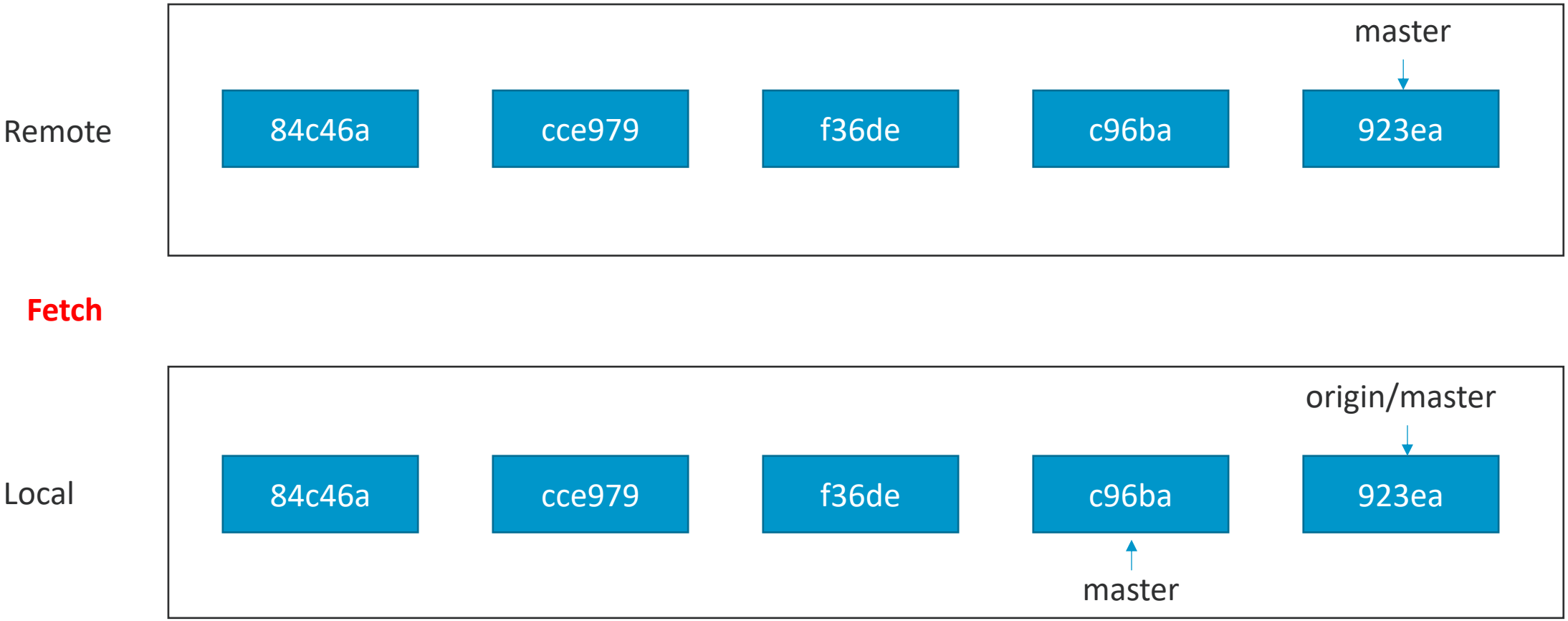
Designed after sample GIF by Kevin Skoglund:
https://www.lynda.com/Git-tutorials/Using-local-remote-repositories/100222/111327-4.html

Using local & remote repositories



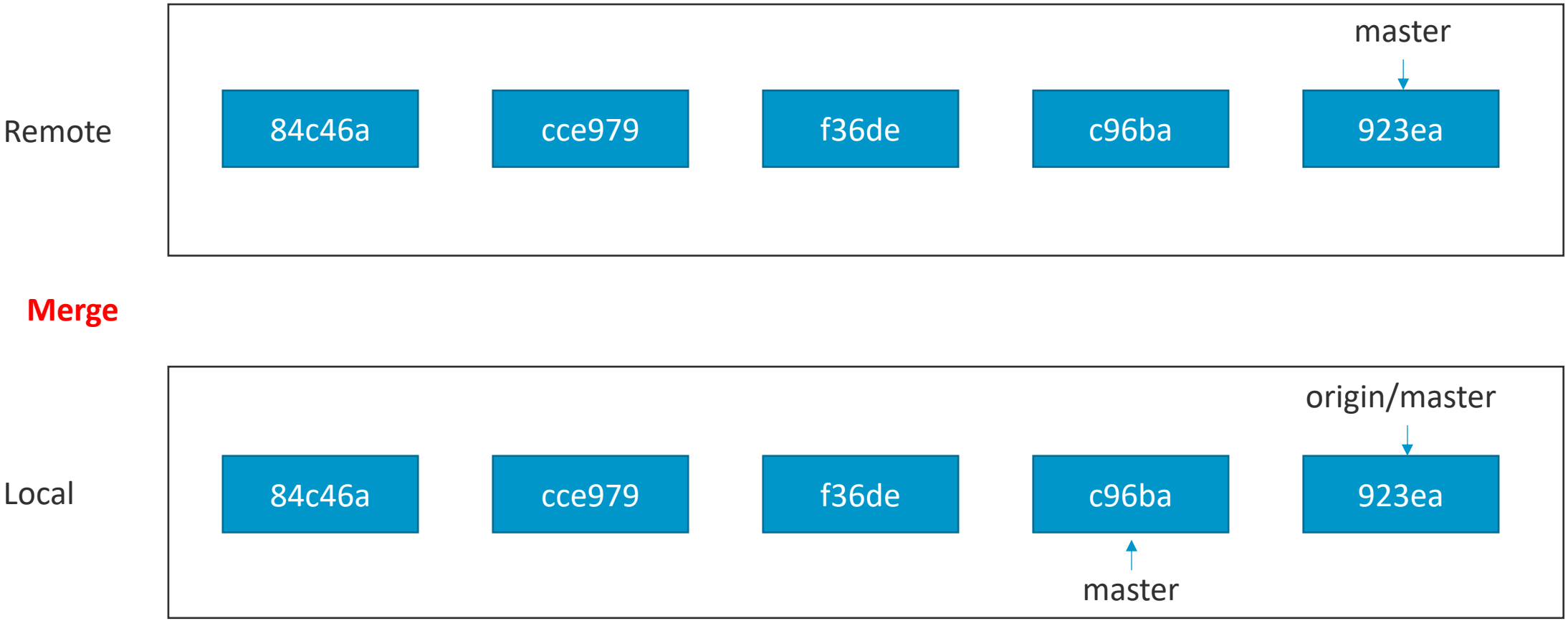
Designed after sample GIF by Kevin Skoglund:
https://www.lynda.com/Git-tutorials/Using-local-remote-repositories/100222/111327-4.html

Using local & remote repositories



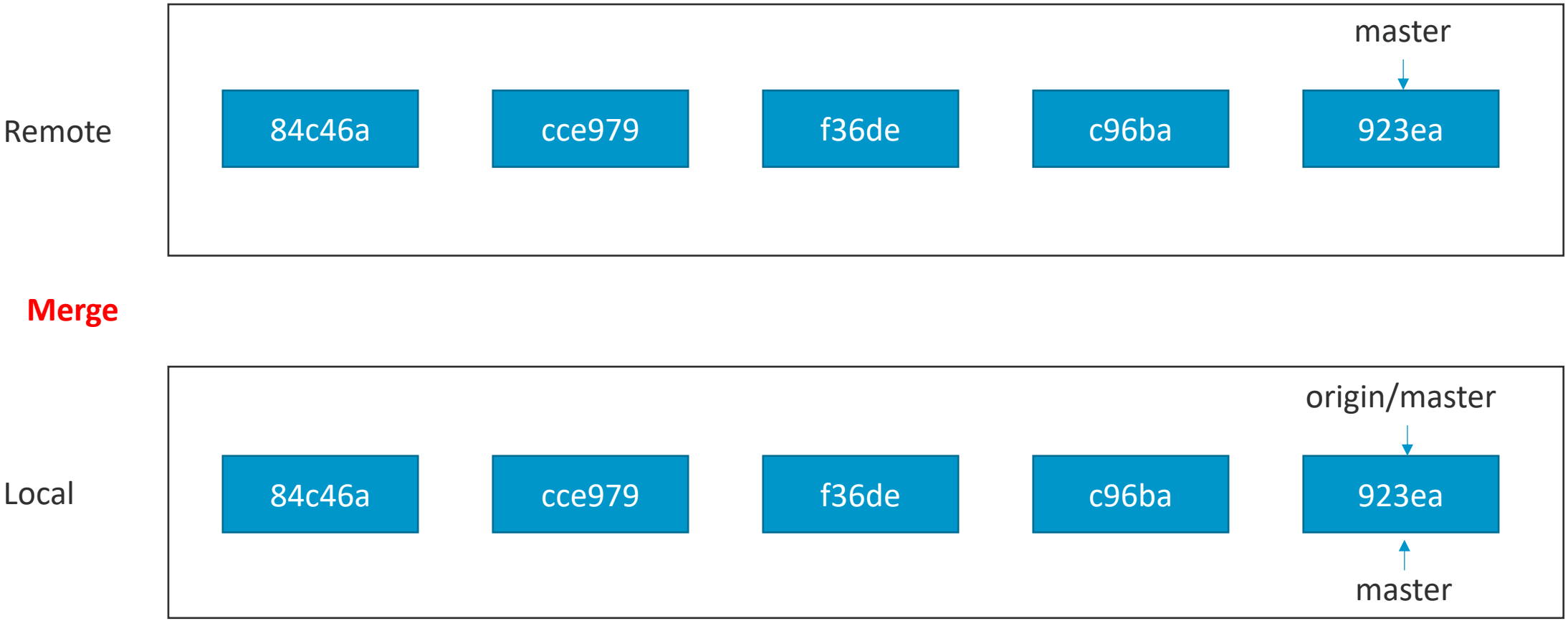
Designed after sample GIF by Kevin Skoglund:
https://www.lynda.com/Git-tutorials/Using-local-remote-repositories/100222/111327-4.html

Using local & remote repositories



Designed after sample GIF by Kevin Skoglund:
https://www.lynda.com/Git-tutorials/Using-local-remote-repositories/100222/111327-4.html

Using local & remote repositories



Designed after sample GIF by Kevin Skoglund:
https://www.lynda.com/Git-tutorials/Using-local-remote-repositories/100222/111327-4.html

Hoe ziet een conflict eruit?

SIGMADATA Learning

```
<<<<<<< HEAD
hoiasdf

Test!
=======
hoi
>>>>>>> refs/remotes/origin/master
```

Git commands - overview

SIGMADATA Learning

https://git-scm.com/docs

Command	Meaning
init	Initialize new repository
clone	Clone a repository into a new directory
add	Add new files from working tree to snapshot
status	Show the working tree status
commit	Record changes in the repository
rm	Remove files from working tree and from index
mv	Move or rename a file
branch	List, create or delete branches
checkout	Switch branches
merge	Join two or more branches
fetch	Download from another repository
pull	fetch + merge
push	Upload to another repository
remote	Manage set of tracked repositories

SIGMADATA
Learning

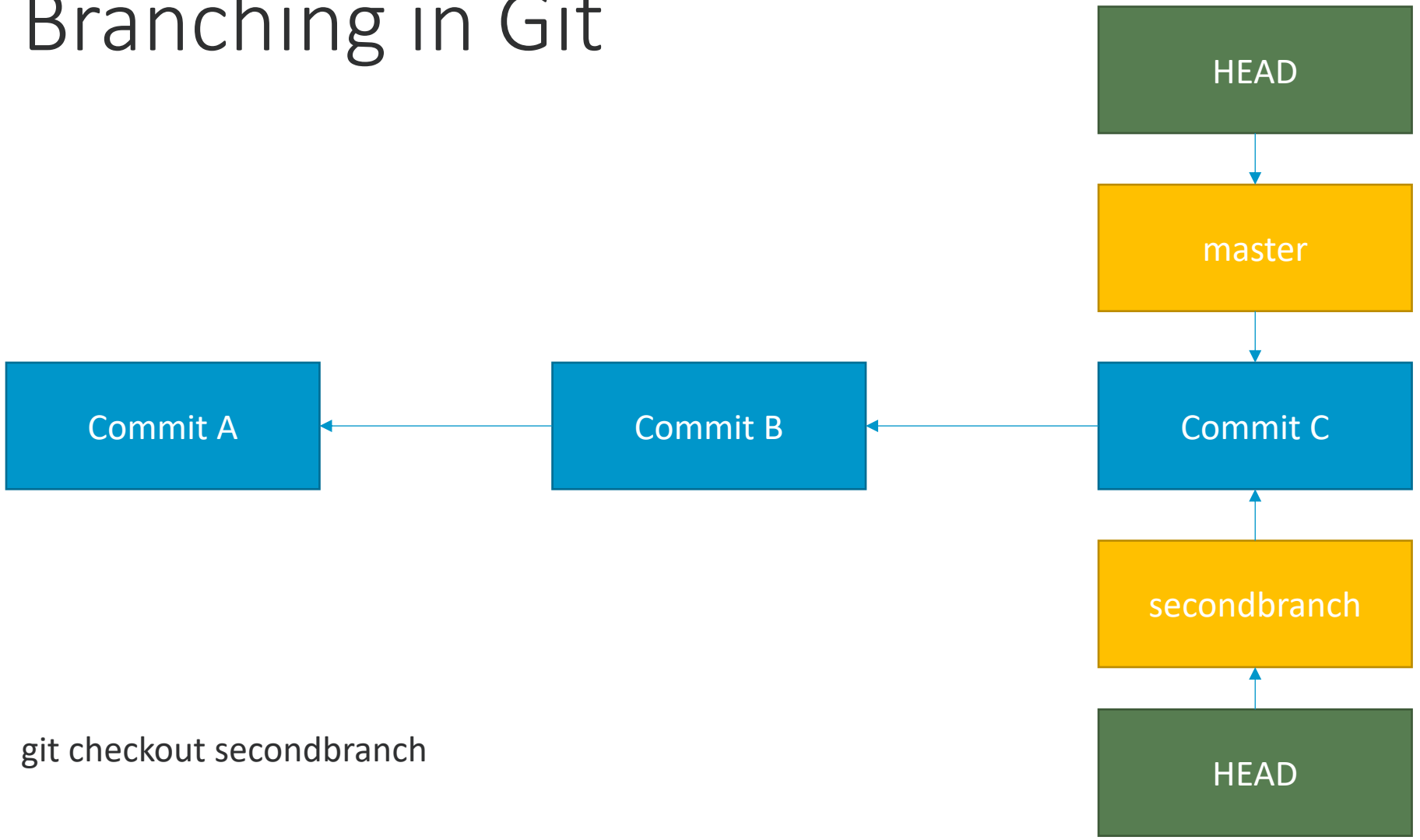
Git branching & merging

Basic Branching

SIGMADATA Learning

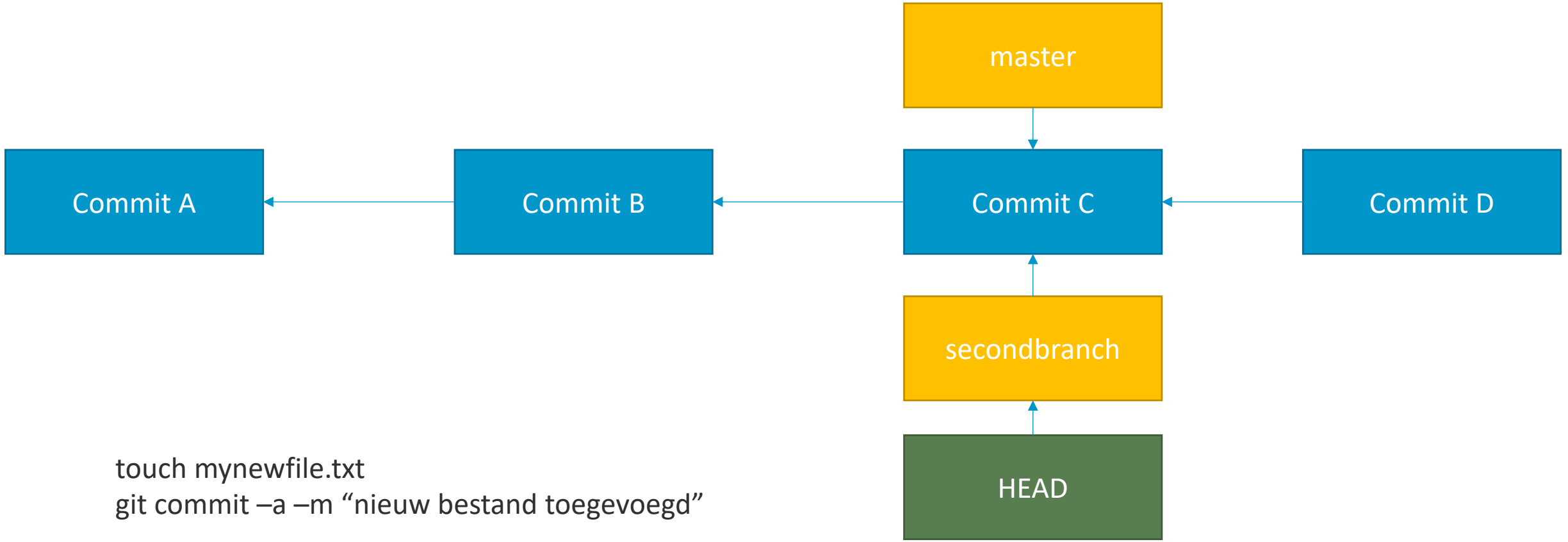
- Branching: *to diverge from the main line of development and continue to do work without messing with that main line*
- Git's "Killer Feature"
  - Lightweight
  - Doesn't copy all underlying files
  - *A branch in Git is simply a lightweight movable pointer*
- Default branch name is master
  - Not a special branch!
  - Just created by default by git init
  - Most people don't bother to change it

Branching in Git



git checkout secondbranch

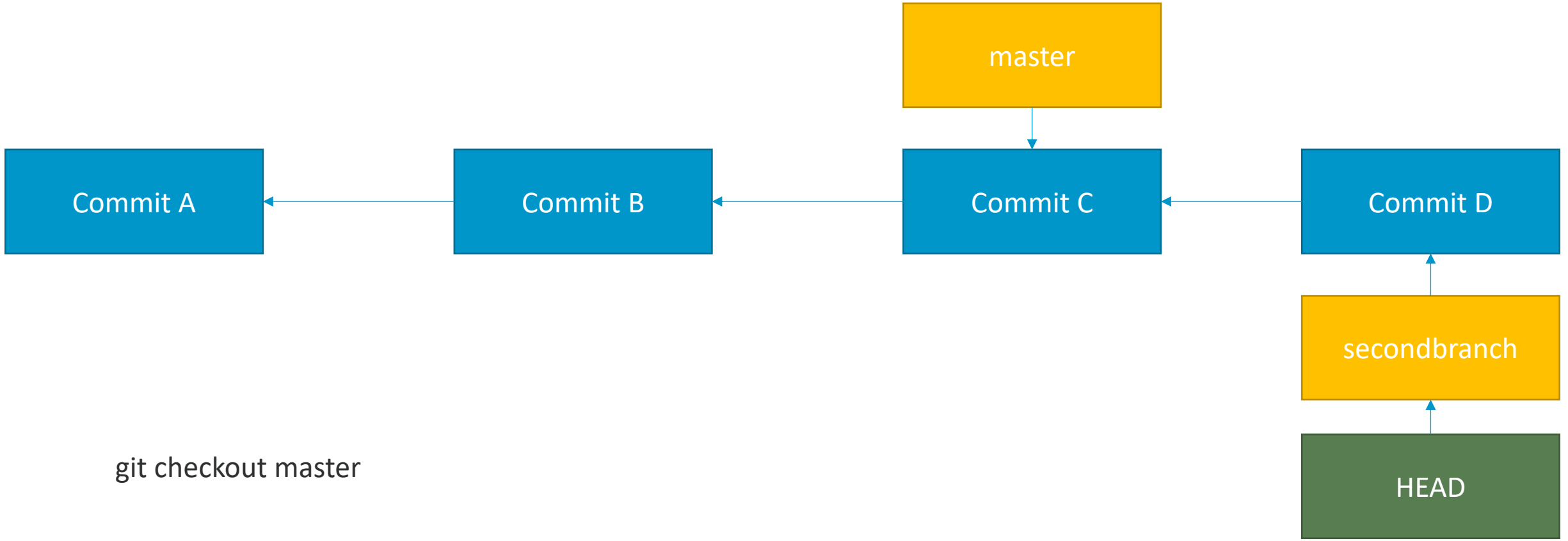
Branching in Git



touch mynewfile.txt
git commit –a –m “nieuw bestand toegevoegd”

Branching in Git

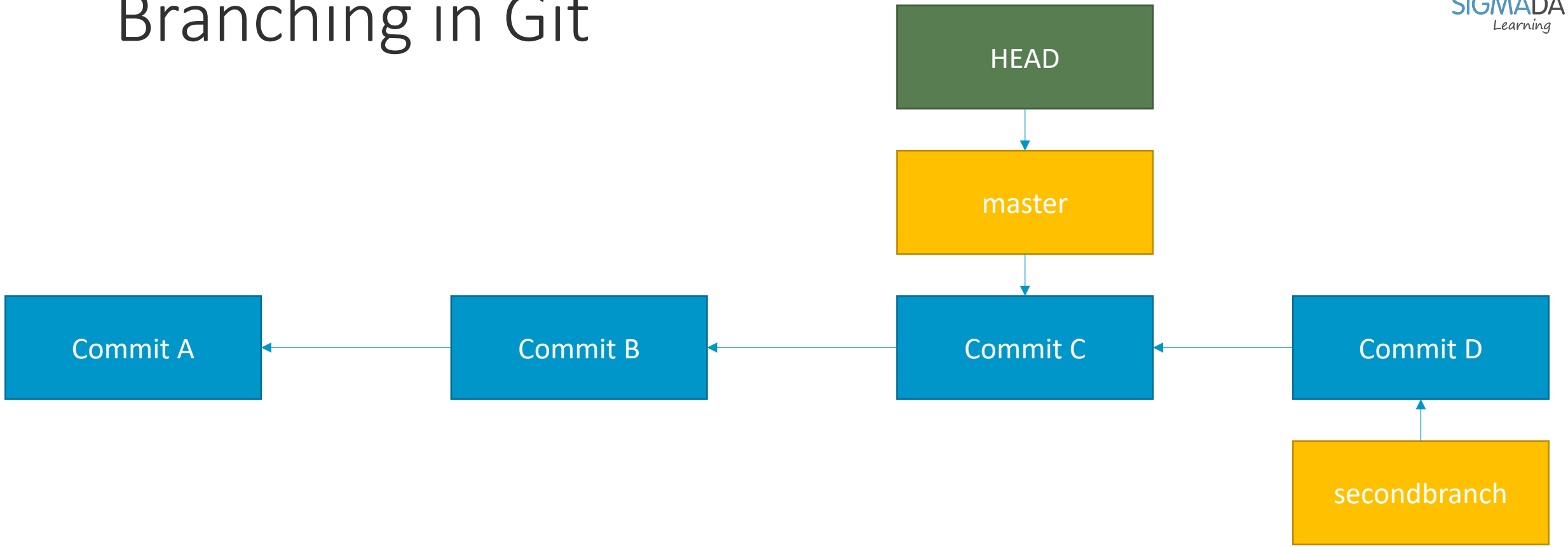
SIGMADATA Learning



git checkout master

Branching in Git

SIGMADATA Learning



git checkout master

Branching in Git – Demo using git-webui

SIGMADATA Learning

- https://github.com/alberthier/git-webui
- init new repository
- git webui
  - add/edit three files (+ commit)
  - git branch mijntestbranch
  - git checkout mijntestbranch
  - Commit on testbranch
  - git checkout master
  - Commit on master
  - Show divergent history in git webui

You can execute these two steps together using:
git checkout –b mijntestbranch

Branching in Git – Demo using git-webui

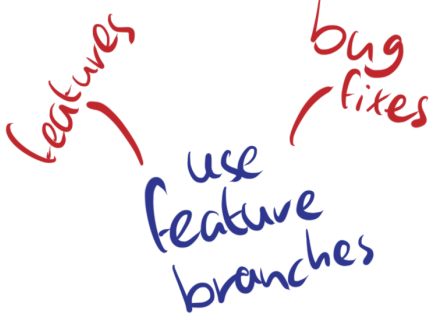
SIGMADATA Learning

- Bug arises in master while developing on test!
  - checkout master
  - branch & checkout “hotfix”
  - commit changes for hotfix
    - Show proceedings in git webui
  - checkout master
  - merge hotfix
    - Show result in git webui
  - Delete branch hotfix
    - git branch –d hotfix

Merging in Git

SIGMADATA Learning

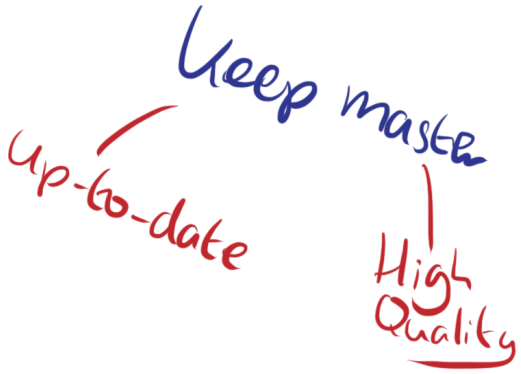
- Basic merge: *three-way*
  - Snapshot to merge into
  - Snapshot to merge in
  - Common ancestor → makes it way easier than Subversion or CVS
- Git creates a new snapshot as the result of this merge

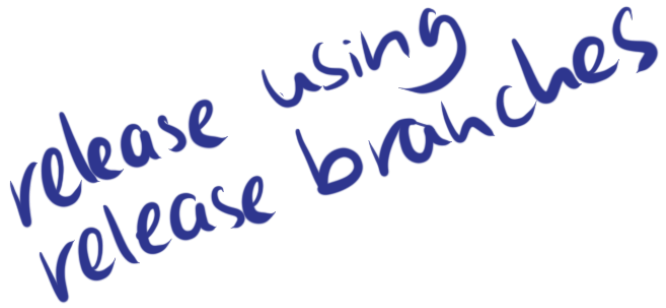


SIGMADATA
Learning

Microsoft about Branch Strategy

https://docs.microsoft.com/en-us/azure/devops/repos/git/git-branching-guidance

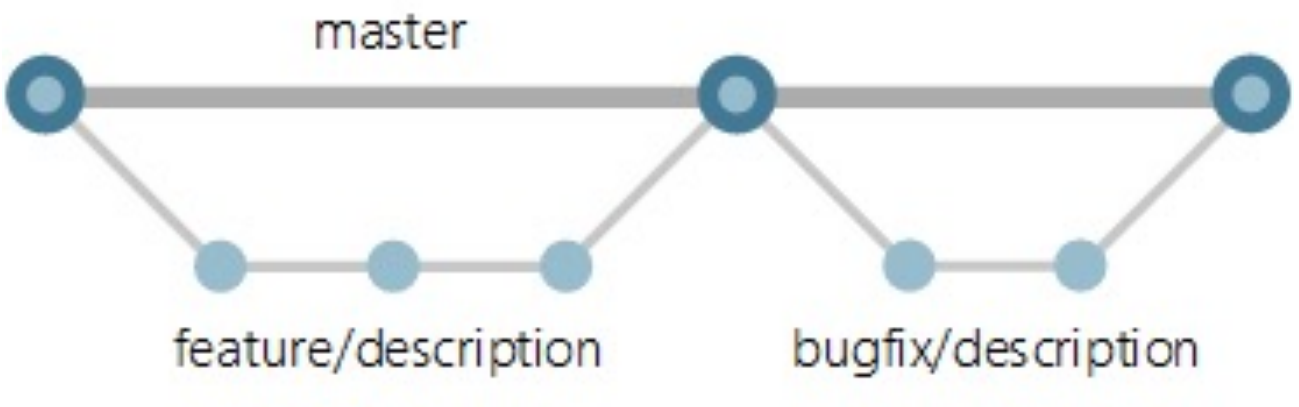


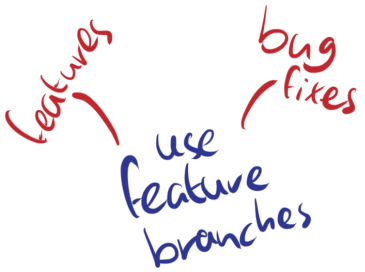


Use Feature Branches

SIGMADATA Learning

`master` is your *collaboration branch*
Don't commit directly to master!



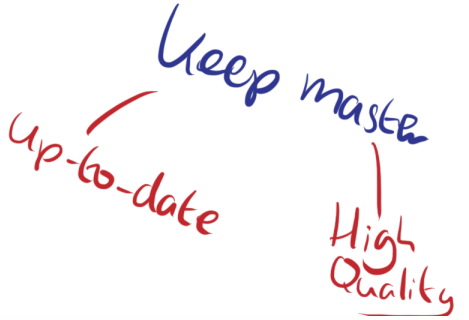


Merge using Pull Request into Master

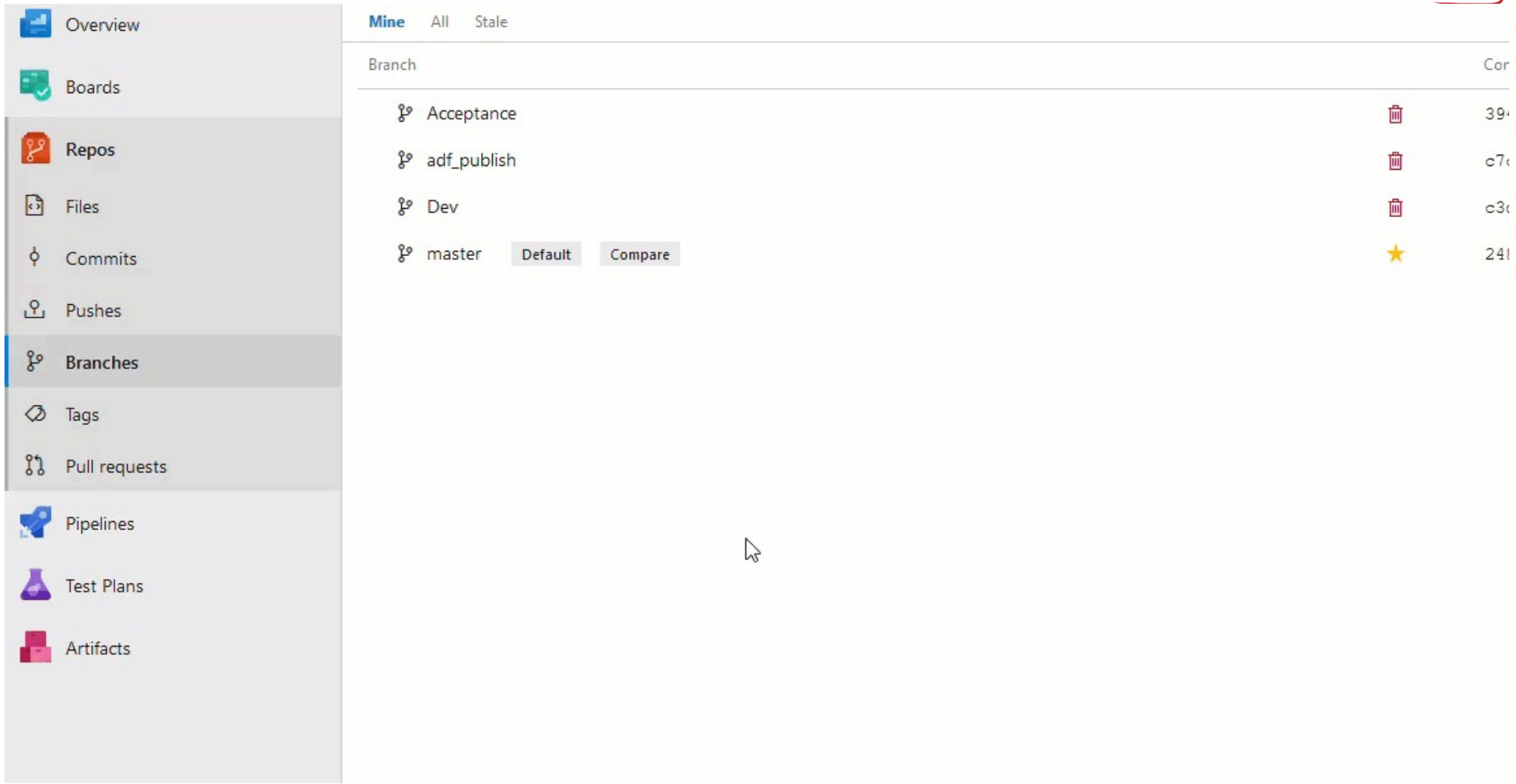
SIGMADATA Learning

- Dev A:
  - “Hey team, please merge this awesome new code into master?”
- Reviewer (Dev B):
  - ~~“Sure, once you fix this ugly piece of shit”~~
  - “Great work A! One small note: please fix this!”
  - Approves (after A’s fix)
- Reviewer (Dev C):
  - No problems found
  - Approves
- **Merge system takes over and merges feature branch to master**

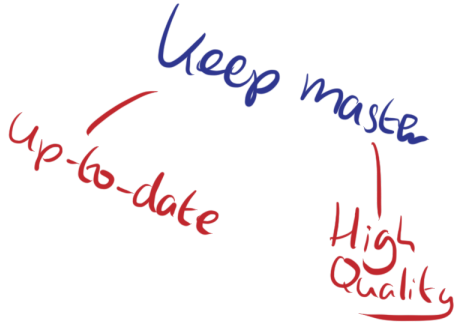
Keep Master High-Quality



SIGMADATA Learning



Keep Master High-Quality



SIGMADATA Learning

Protect this branch

- Setting a Required policy will enforce the use of pull requests when updating the branch
- Setting a Required policy will prevent branch deletion
- Manage permissions for this branch on the Security page

☑ **Require a minimum number of reviewers**

Require approval from a specified number of reviewers on pull requests.

Minimum number of reviewers: 2

☐ ~~Requestors can approve their own changes~~

☐ Allow completion even if some reviewers vote to wait or reject

☐ Reset code reviewer votes when there are new changes

☐ **Check for linked work items**

Encourage traceability by checking for linked work items on pull requests.

☐ **Check for comment resolution**

Check to see that all comments have been resolved on pull requests.

☑ **Limit merge types**

Control branch history by limiting the available types of merge when pull requests are completed.

Allowed merge types:

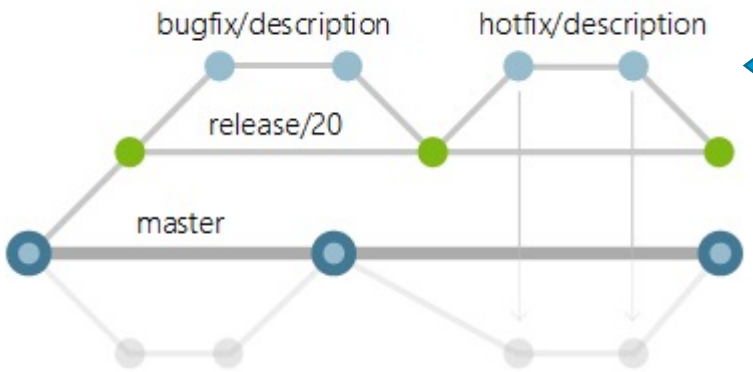
Creates a semi-linear history by replaying the source branch commits onto the target and then creating a merge commit.

Build validation

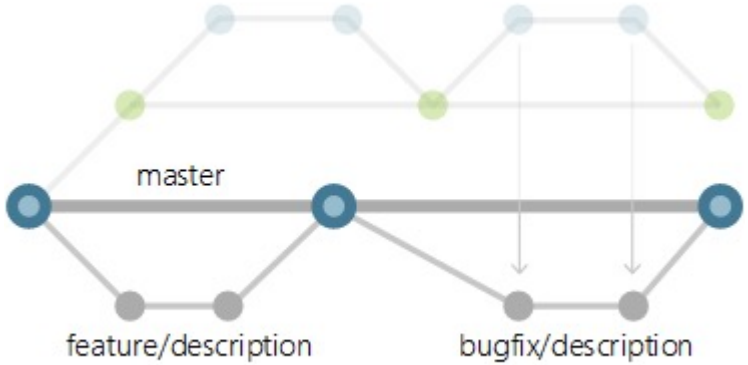
Validate code by pre-merging and building pull request changes

+ Add build policy

Release using Release Branches



Use cherry-picking to bring changes to a **new feature branch**

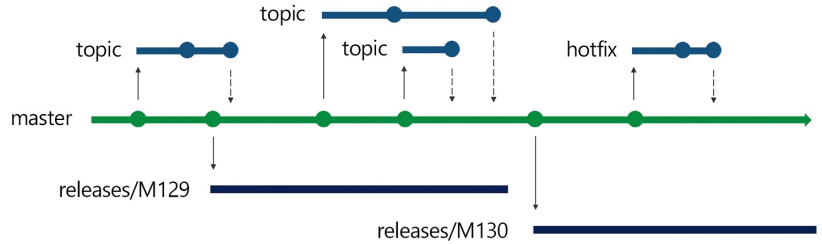


release using release branches

Release Flow Branching Structure

SIGMADATA Learning

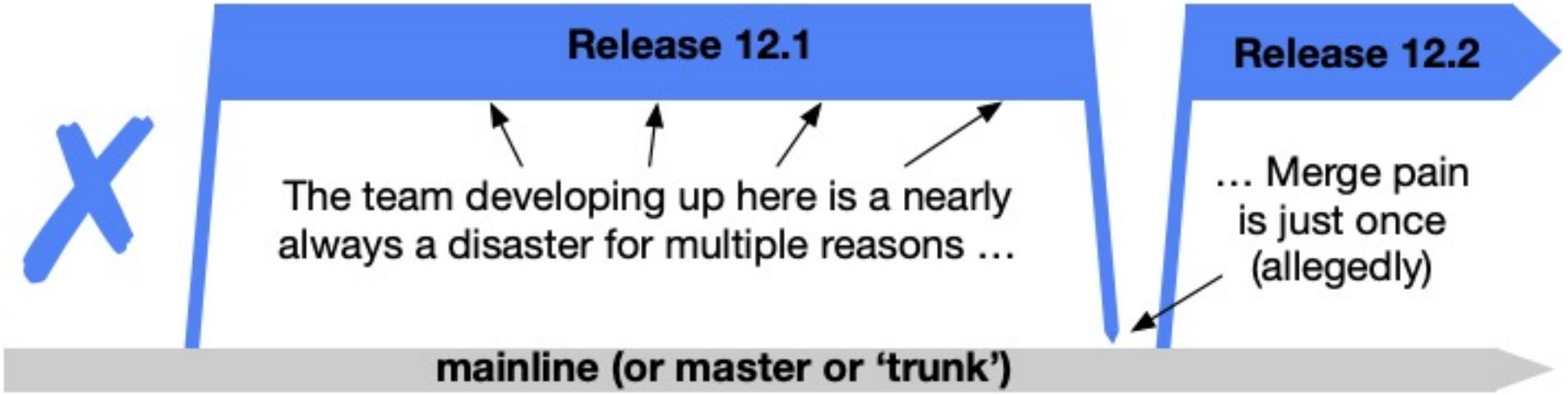
- Releases are *sprintly* (every X weeks)
  - Branch accordingly (BUT trunk-based!! https://trunkbaseddevelopment.com/)
- Features / topics within topic branches
- Bugfixes merged into master first
  - Then cherrypicked into release
- Every release goes through master first



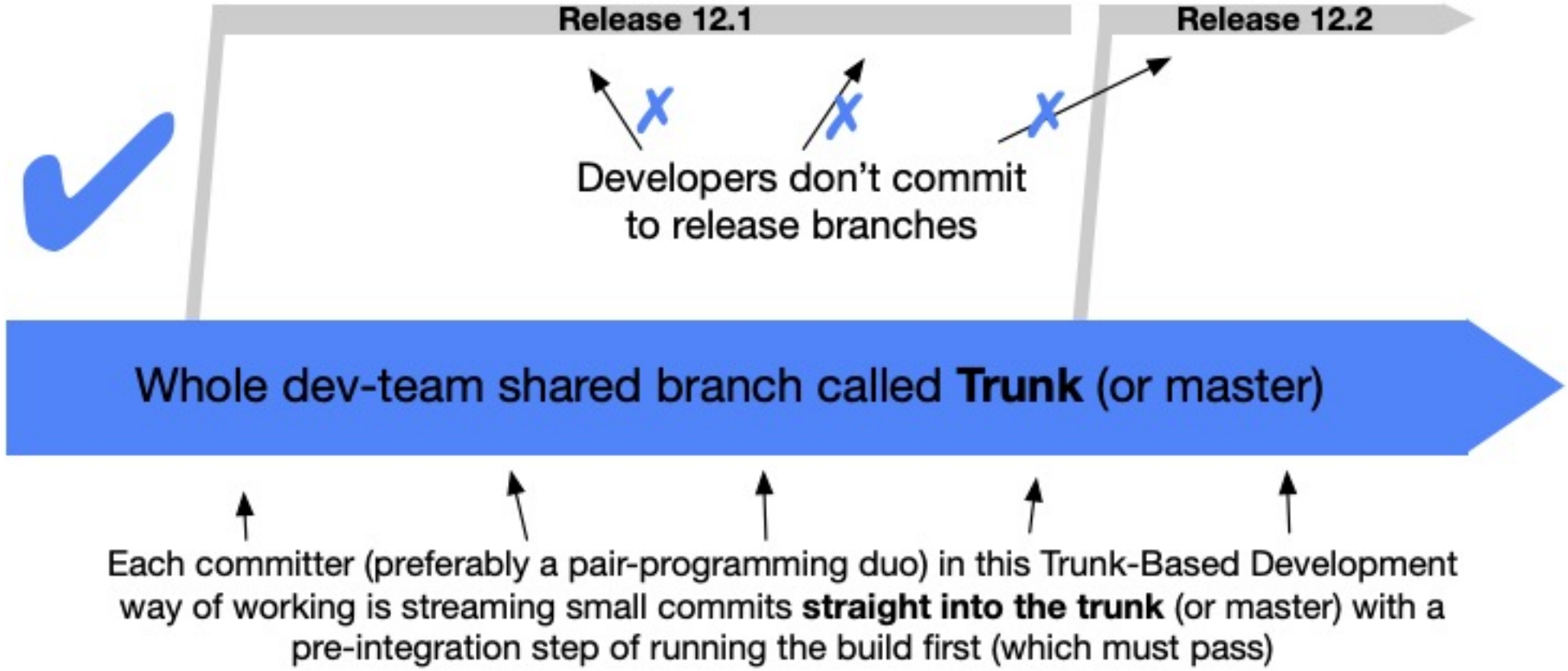
https://devblogs.microsoft.com/devops/release-flow-how-we-do-branching-on-the-vsts-team/

Don’t: develop in release branches, then merge back

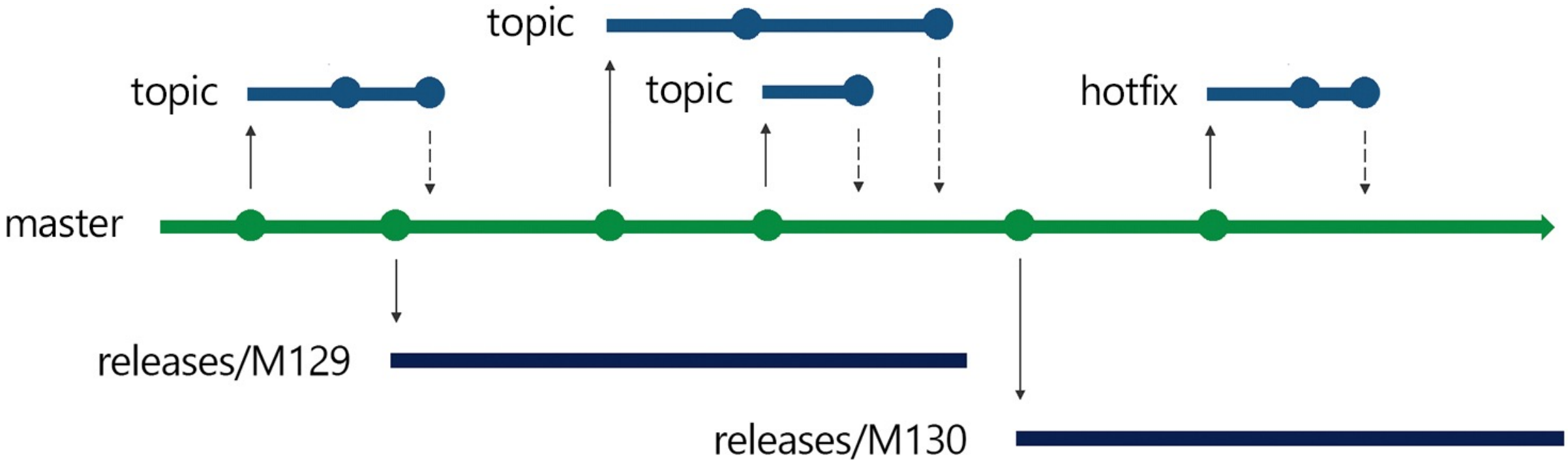
SIGMADATA Learning



Do: *trunk-based development*



Release Flow Branching Structure



https://devblogs.microsoft.com/devops/release-flow-how-we-do-branching-on-the-vsts-team/

Course Overview (day 2)

SIGMADATA
Learning

- Setting up projects
- Setting up build & release pipelines
- Building and deploying BI projects
- Test automation

SIGMADATA
Learning

Setting up projects

Azure DevOps for BI & Data Platforms

Projects: single or multiple?

SIGMADATA Learning

Single project	Multiple projects
All work in one ”portfolio”	Shifts administration burden
Share source repos, build/release def, etc.	Git makes it easy to move repos between projects
Good for large products managed by multiple teams with tight interdependencies	Alignment challenges
Teams have visibility in each other’s work -> alignment	Greater control of security and access to assets
Teams use same taxonomy for work item tracking	More autonomy to manage project
Sample: Azure DevOps • 40 feature teams • 500 users • single project	

SIGMADATA
Learning

Setting up Azure Pipelines

Azure DevOps for BI & Data Platforms

Azure Pipelines

SIGMADATA
Learning

- Used for
  - CI
  - CD
- Multi-stage
- Defined using YAML

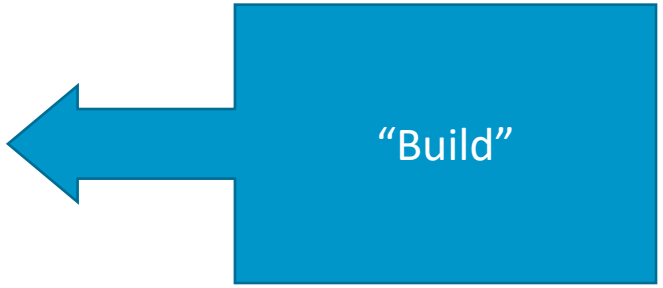
Pipeline structure

SIGMADATA Learning

- Pipeline Stage A
  - Job 1
    - Step 1.1
    - Step 1.2
    - …
  - Job 2
    - Step 2.1
    - Step 2.2
    - …
- Stage B
  - …

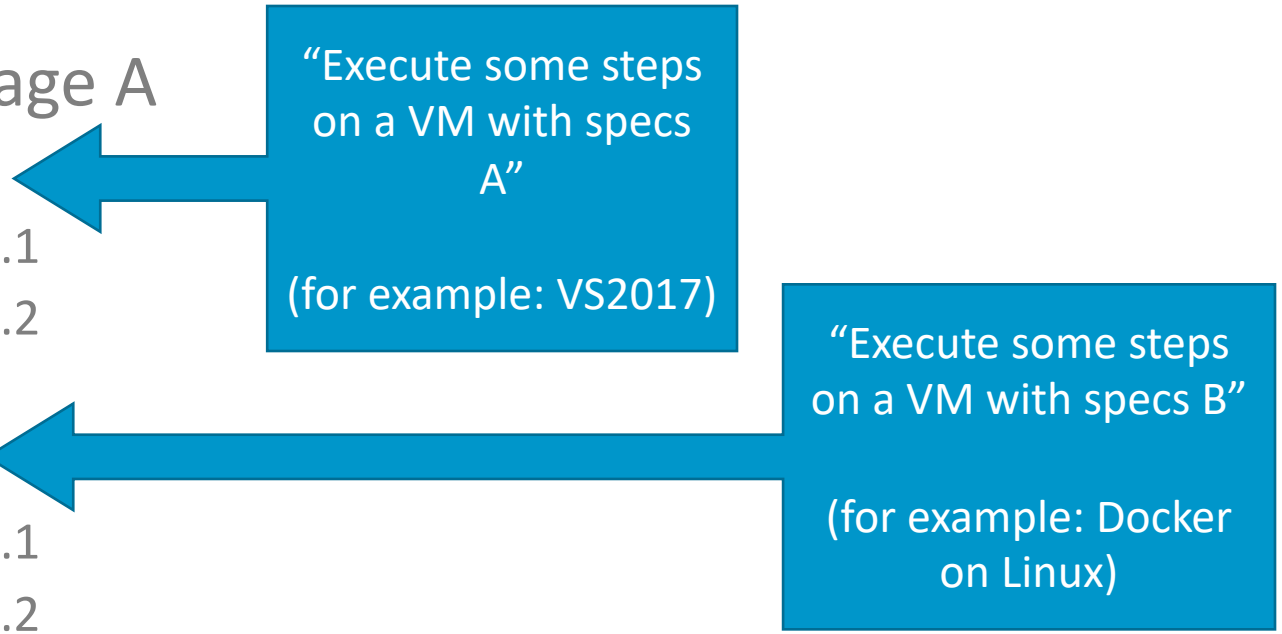
Pipeline structure - *stages*

- Pipeline Stage A
  - Job 1
    - Step 1.1
    - Step 1.2
    - …
  - Job 2
    - Step 2.1
    - Step 2.2
    - …
- Stage B
  - …



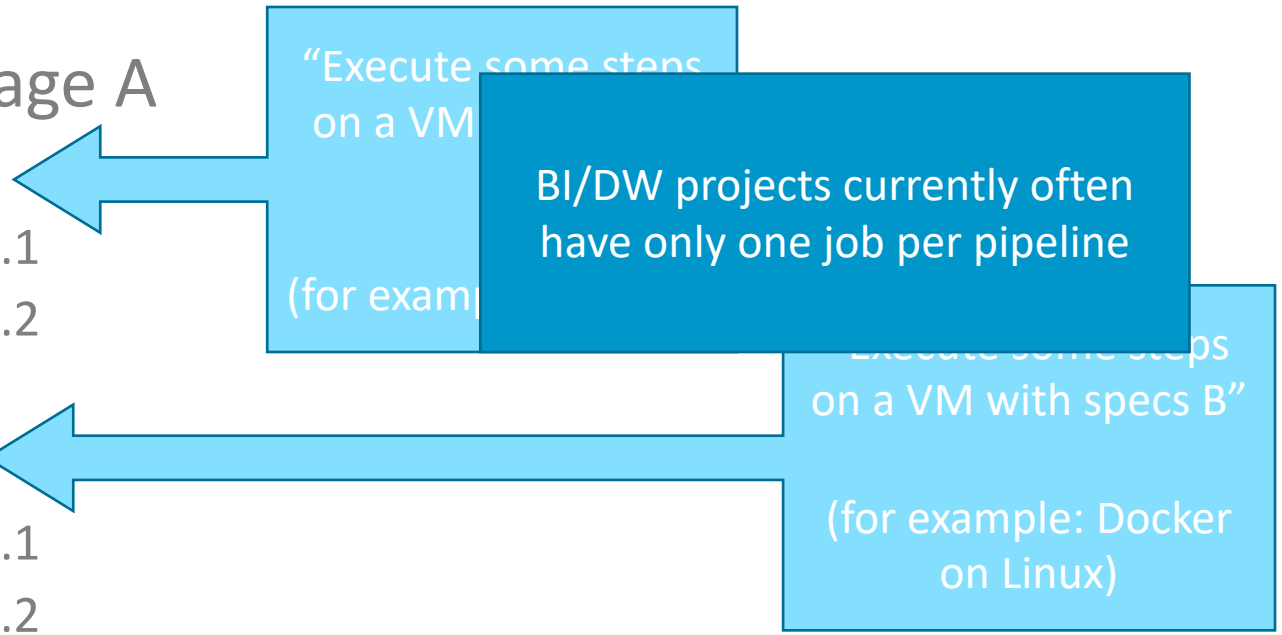
Pipeline structure - *Jobs*

- Pipeline Stage A
  - Job 1
    - Step 1.1
    - Step 1.2
    - …
  - Job 2
    - Step 2.1
    - Step 2.2
    - …
- Stage B
  - …



Pipeline structure - *Jobs*

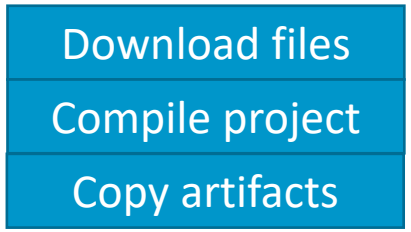
- Pipeline Stage A
  - Job 1
    - Step 1.1
    - Step 1.2
    - …
  - Job 2
    - Step 2.1
    - Step 2.2
    - …
- Stage B
  - …



Pipeline structure - *steps*

SIGMADATA Learning

- Pipeline Stage A
  - Job 1
    - Step 1.1
    - Step 1.2
    - ...
  - Job 2
    - Step 2.1
    - Step 2.2
    - ...
- Stage B
  - ...



Sample of a YAML-pipeline (1)

SIGMADATA Learning

YAML

```yaml
stages:
- stage: Build
  jobs:
  - job: BuildJob
    steps:
    - script: echo Building!
- stage: Test
  jobs:
  - job: TestOnWindows
    steps:
    - script: echo Testing on Windows!
  - job: TestOnLinux
    steps:
    - script: echo Testing on Linux!
- stage: Deploy
  jobs:
  - job: Deploy
    steps:
    - script: echo Deploying the code!
```

Sample of a YAML-pipeline (2)

SIGMADATA Learning

```
# Build Pipeline for SQL Database

# De *trigger* geeft aan waardoor deze pipeline automatisch
geval: een *commit* tegen *master*
trigger:
- master

# De *pool* geeft aan van wat voor VM-image gebruik gemaakt moet worden om deze pipeline
uit te kunnen voeren. In dit geval dus een image met Visual Studio 2017
pool:
  name: Hosted VS2017
  demands:
  - msbuild
  - visualstudio

# Hier volgende daadwerkelijke "build-stappen". In dit geval hebben we een "eenvoudige"
build pipeline - later zullen we kijken naar pipelines met meerdere "stages".
steps:
```

If you only have one stage, you can omit the ”stage” element

If you only have one stage AND you only have one job, you can omit the “job” element as well!

Module 7 - Lab

SIGMADATA Learning

- Create your own pipeline!
  - Build
  - “advanced options” for build
  - Multi-stage
- After that: round-up
  - Environments
  - Pre-deployment approvals

Deployments with YAML

SIGMADATA Learning

- Besides the “job” also the “deployment”
- Environment
  - Approvals
  - Resource types (currently “Kubernetes”, “Virtual Machine” or “None”)
  - ~~Future~~ Today:
    - Checks (gated deployments)
    - Other environment types (”provision server according to these specs”)

Deployments with YAML - demo

SIGMADATA Learning

- Creating an environment
- Using a “deployment” instead of a “job”
- Setting up an approval

Multi-stage pipelines – what's next

SIGMADATA Learning

In addition to the preview features that now available, there are so many exciting things just around the corner for Azure Pipelines we want to share:

- **Caching** – We'll be announcing the availability of another much-requested feature very shortly: caching to help your builds run even faster.
- ~~**Checks and approvals** – We're improving multi-stage pipelines with the ability to set approvals on your environments, to help control what gets deployed when and where. We'll keep iterating here to deliver more experiences with checks to help gating your multi-stage pipelines.~~
- **Deployment strategies** – We're adding additional deployment strategies as part of the deployment job type, like blue-green, canary and rolling, to better control how your applications are deployed across distributed systems.
- **Environments** – We're adding support for additional resource types in environments, so you can get going quickly with virtual machines (through deployment groups) and Azure Web Apps.
- **Mobile** – With our new UX, we're going to start to enable new mobile views in Q2 to help you view the status of pipelines, quickly jump into logs, and complete approvals.
- **Pipeline analytics** – We're continuing to grow our pipeline analytics experiences to help you get an all-up picture of the health of your pipelines, so you can know where to go in and dig deeper.
- **Tests and code coverage** – We're going to be shipping all new test and code coverage features and UX in the next months.

https://devblogs.microsoft.com/devops/whats-new-with-azure-pipelines/

YAML pipelines - roundup

SIGMADATA Learning

- YAML is default for build pipelines
- YAML will be default for release pipelines as well
  - Easier version control, sharing pipeline definitions, etc.
  - Currently in preview

SIGMADATA
Learning

Build and release BI projects

Azure DevOps for BI & Data Platforms

Build

Integrating all developed code in one shared repo.

Build it to create artifacts.

Build

- Conceptually easy: just create artifacts
- Reality:
  - Some projects deploy out-of-the-box
  - Some projects don’t
- Be prepared to split your build into multiple steps by project type!

Build: Folder structure

- Group by project type in folders
  - SSIS\…
  - SSAS-TM\…
  - SSAS-MD\…
  - Etc.
- (or the Git way: give each ”independently deployable” its own repository)
- Use one solution per project (or at most one solution per project type) inside these folders
  - SSIS\MySSISProject.sln
  - SQL\MySQLDatabaseProject.sln
  - Not: SQL\AllProjectsInOneSolution.sln*
- This enables build & deployment per project type

* You can create other solutions as well – just store them outside these folders

Sample solution structure

- SQL Database as DW
- Storage Account as landing zone (V2)
- Storage Account as staging area (BlobStorage)
- ADF for ETL

Workflow – SQL Database

SIGMADATA Learning

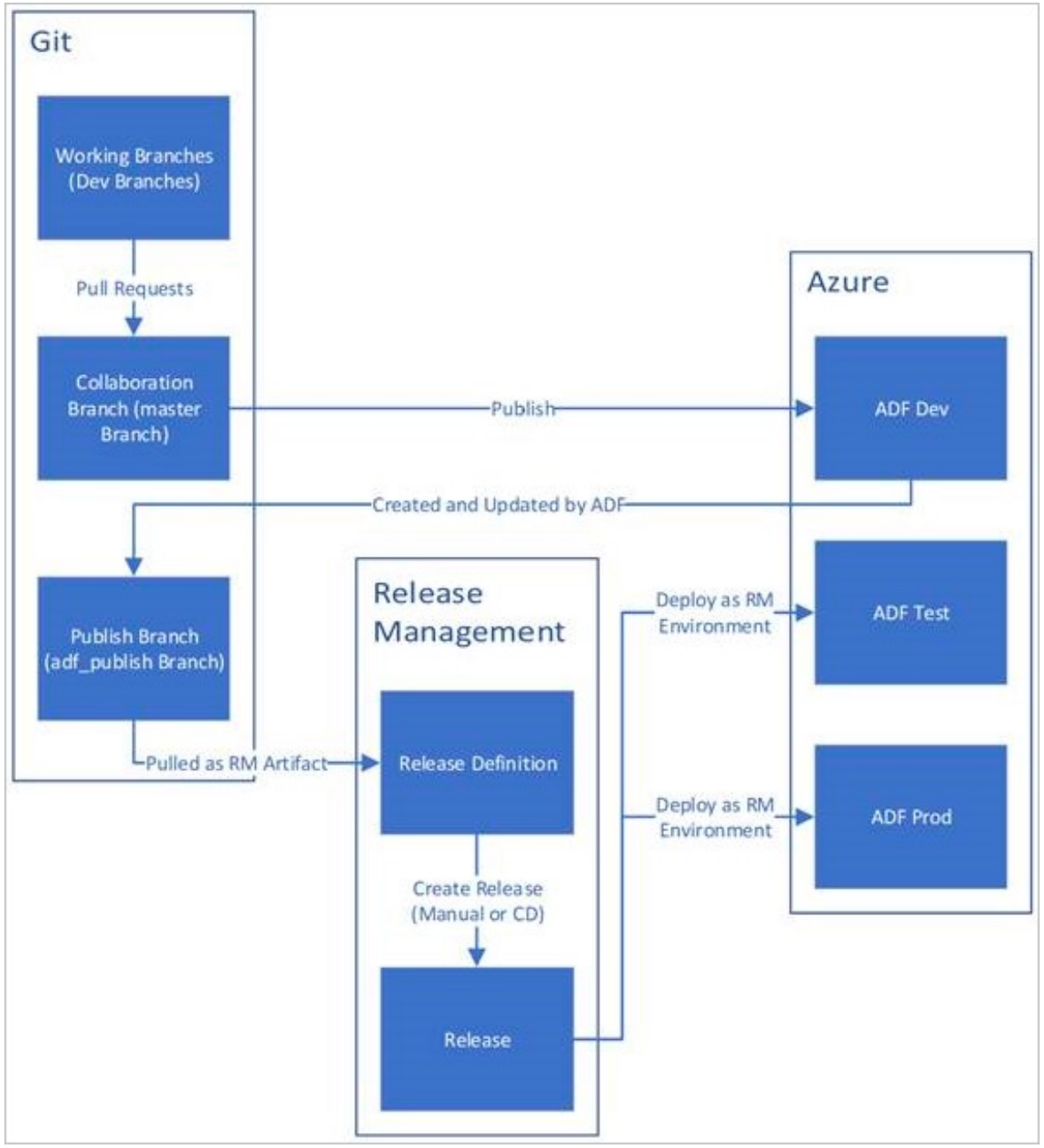
- Database project in Visual Studio
- CI: MSBuild natively supported on Azure DevOps Build Pipeline
- CD: Deployment via native SQL Database Deploy task

Workflow – Azure Data Factory

SIGMADATA Learning

- Set up a development repo / branch in Azure Repos
- Test your changes with Debug
- Create pull request to master
- Master-publish using ADF's **publish**
  - No real "artifact" here -> the ARM definition *is* the artifact
- Release using Azure DevOps Release Pipeline
  - Use **Azure Resource Group Deployment**

Workflow – ADF



Source: https://docs.microsoft.com/en-us/azure/data-factory/continuous-integration-deployment#best-practices-for-cicd

Integration Runtime considerations

- SSIS-IR: not visible in Azure Portal
  - Is a VM running SSIS runtime
  - Can be paused within ADF (or via script). Takes +/- 6 minutes!!
  - Can be customized
    - https://docs.microsoft.com/en-us/azure/data-factory/how-to-configure-azure-ssis-ir-custom-setup
    - In short:
      - prepare a setup script that will be executed every time IR is reimaged / started
      - Write this to a storage account
      - Configure the SSIS-IR to use the “Custom Setup Container SAS URL” using ADF (or via script)
      - **Use a v3 instance – v2 is not supported for custom setup!**

SSIS

SIGMADATA Learning

- MSBuild doesn’t support SSIS projects
  - There is a workaround with a generic “.proj” file
    - Create a generic “.proj” file containing logic to call on the library
    - Use MSBuild to build this “.proj” file like all other files
  - http://www.msbiblog.com/2017/01/31/vsts-continuous-msbuild-for-dwh-bi/
- Custom component from Azure DevOps Marketplace
  - SSIS DevOps Tools (Microsoft, Preview)
  - SSISBuild
  - Other custom components
- Hand-crafting an *.ispac*
  - http://www.msbiblog.com/2018/05/29/building-ssis-projects-in-visual-studio-team-services/

SSAS-MD:

- Basically: create custom MSBuild projects
- http://www.dimodelo.com/blog/2014/automate-build-and-deployment-of-ssas-cubes-using-msbuild/
- http://t-sql.dk/?p=1724

SSRS

- What do you want to “build”?
- Two options
  - Copy “naïvely” all reports, datasets, datasources
  - Traverse rptproj file (XML)

Getting build to work - Roundup

- In general, three options:
  1. Build works out of the box (SQL Server, SQL Database, SSAS-TM)
  2. Build using custom components
     - Hand-written
     - External components (from Azure DevOps marketplace, NuGet etc.)
  3. Build using hand-written msbuild instructions
  4. (Dev tooling is integrated with the CI process --> Azure Data Factory v2 is!)
- Easier to tackle if you split out folders by project type
- Remember to gather all artifacts and publish 'em ☺

Release

Ensure software can reliably be released.
At any time.

SQL Server

SIGMADATA Learning

- Use the default Database Deploy task

SSIS

SIGMADATA Learning

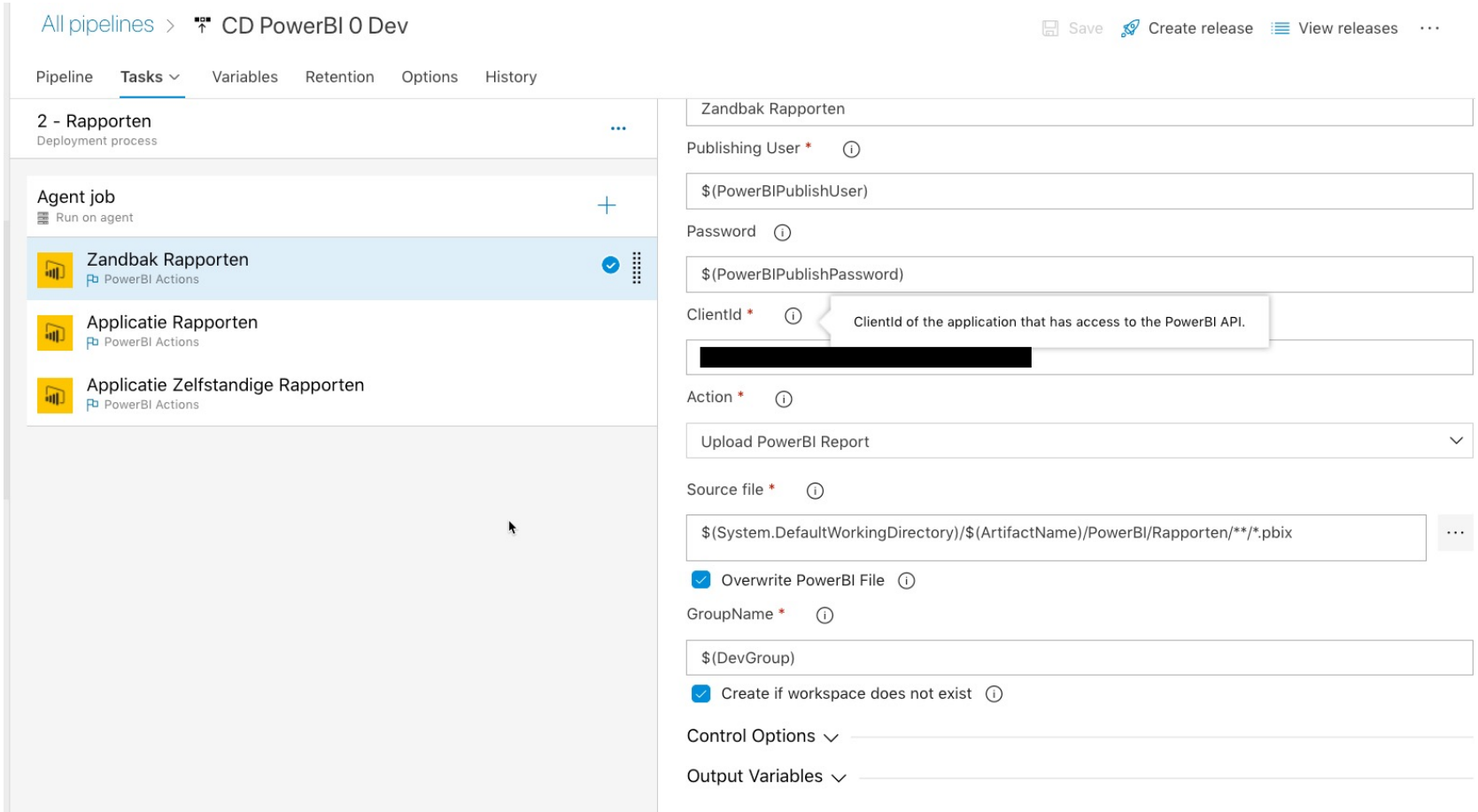
- SSIS DevOps Tools (Marketplace, *in preview*)
- SSISBuild (includes ssisdeploy.exe)
- SSIS Library in Powershell
  - [System.Reflection.Assembly]::LoadWithPartialName("Microsoft.SqlServer.Management.IntegrationServices")
  - https://docs.microsoft.com/en-us/sql/integration-services/ssis-quickstart-deploy-powershell?view=sql-server-2017
- isdeploymentwizard.exe

SSAS-TM

SIGMADATA Learning

- Use Microsoft.AnalysisServices.Deployment.exe
- Part of SSMS installation (install on the build agent)
- Configuration via three files:
  - .configsettings -> data source connection strings
  - .deploymentoptions -> role deployment, partition deployment etc.
  - .deploymenttargets -> AS server to deploy to
- These files can be different <u>per environment</u>

Power BI


All pipelines
CD PowerBI 0 Dev
Save
Create release
View releases
Pipeline
Tasks
Variables
Retention
Options
History
2 - Rapporten
Deployment process
Agent job
Run on agent
Zandbak Rapporten
PowerBI Actions
Applicatie Rapporten
PowerBI Actions
Applicatie Zelfstandige Rapporten
PowerBI Actions
Zandbak Rapporten
Publishing User *
$(PowerBIPublishUser)
Password
$(PowerBIPublishPassword)
ClientId *
ClientId of the application that has access to the PowerBI API.
Action *
Upload PowerBI Report
Source file *
$(System.DefaultWorkingDirectory)/$(ArtifactName)/PowerBI/Rapporten/**/*.pbix
Overwrite PowerBI File
GroupName *
$(DevGroup)
Create if workspace does not exist
Control Options
Output Variables

SIGMADATA
Learning

Integrating Test Automation

Azure DevOps for BI & Data Platforms

Test automation for BI

SIGMADATA Learning

- What kind of tests?
  - ETL verification
    - Mapping
    - Verification of data types
    - Transformation check
  - Staging data
    - Are duplicates eliminated?
    - Reconciliation check (“row counts”)
  - Data loading
  - Report contents

Database unit tests vs. DW / BI tests

SIGMADATA Learning

- Database unit tests:
  - Like OO
  - Test effects of procedures & functions
  - tSQLt - https://tsqlt.org/ (supported by Redgate)
  - DbFit - http://dbfit.github.io/dbfit/ (on top of Fitnesse; Open Source)
  - DB Test Driven (http://www.dbtestdriven.com/)
- DW / BI tests:
  - NBi (http://www.nbi.io)
  - LegiTest (https://www.sentryone.com/products/pragmatic-workbench/legitest)
  - CrossTest (http://www.x-test.nl/GettingStarted/)

Introduction to NBi

SIGMADATA Learning

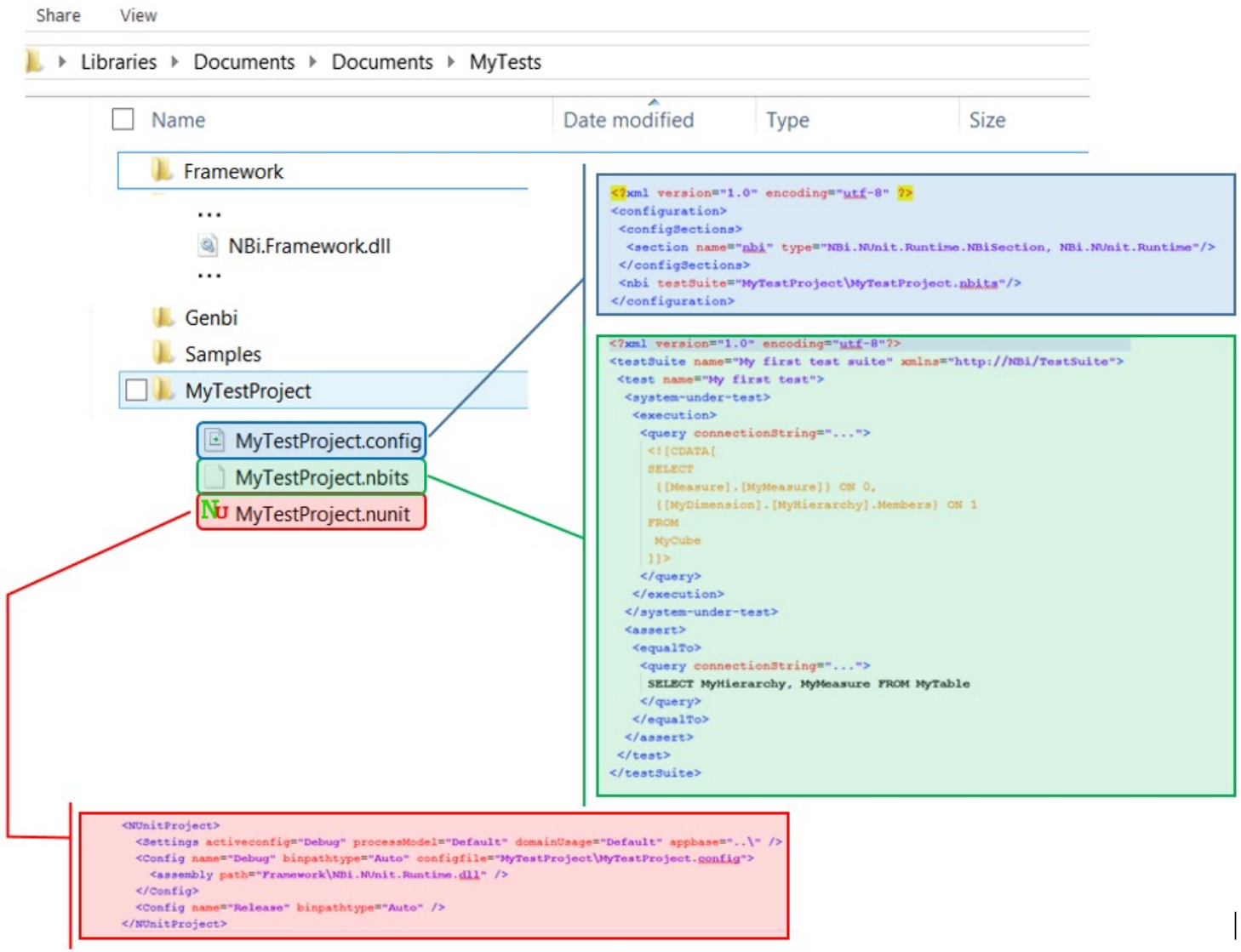
- Framework for testing BI solutions
- Scope
  - Relational databases
  - OLAP / cubes
  - ETL
  - Reports
  - NoSQL databases
- Add-on for Nunit
  - Create tests based on XML

Using NBi

SIGMADATA Learning

- Install using NuGet or setup manually
- Minimal config for test: create three files
  - .config
  - .nbits
  - .nunit

.config, .nbits, .nunit



.nbits

SIGMADATA Learning

- Actual test suite
- XSD available
  - TestSuite
    - Test (within test suite, multiple tests can be defined)
      - System under test
        - Execute something on system
      - Assertion
        - Execute a query / define an explicit assumption


```
<?xml version="1.0" encoding="UTF-8"?>
<testSuite xmlns="http://NBi/TestSuite" name="My first test suite">
  <test name="My first test">
    <system-under-test>
      <execution>
        <query connectionString="...">
          <![CDATA[SELECT {[Measure].[MyMeasure]} ON 0, {[MyDimension].[MyHierarchy].Members} ON 1 FROM MyCube]
        </query>
      </execution>
    </system-under-test>
    <assert>
      <equalTo>
        <query connectionString="...">SELECT MyHierarchy, MyMeasure FROM MyTable</query>
      </equalTo>
    </assert>
  </test>
</testSuite>
```

.nunit

SIGMADATA Learning

- Contains information about:
  - App base – all paths will be relative from here
  - Config file – Relative path to the .config file
  - Assembly path – Relative path to the NBi framework

```
<?xml version="1.0" encoding="UTF-8"?>
<NUnitProject>
  <Settings activeconfig="Default" processModel="Default" domainUsage="Default" />
  <Config name="Default" binpathtype="Auto" appbase="..\" configfile="MyTestProject\MyTestProject.conf
    <assembly path="Framework\NBi.NUnit.Runtime.dll" />
  </Config>
</NUnitProject>
```

.config

SIGMADATA Learning

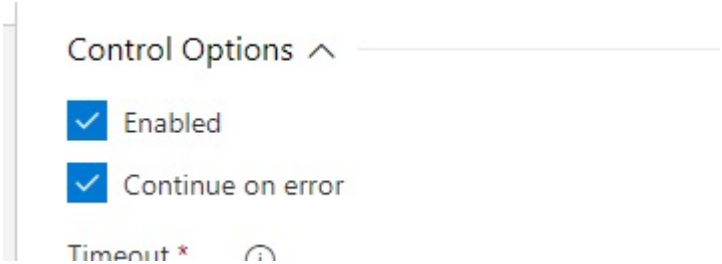
- Defines the path to your *test suite* (.nbits)
- Relative to the *appbase* in the *.nunit* file

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <configSections>
    <section name="nbi" type="NBi.NUnit.Runtime.NBiSection, NBi.NUnit.Runtime" />
  </configSections>
  <nbi testSuite="MyTestProject\MyTestProject.nbits" />
</configuration>
```

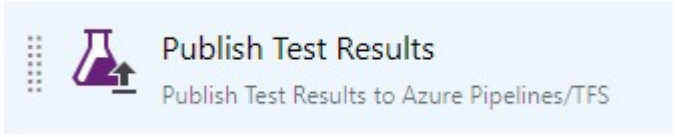
Integrating NBi in Azure Pipelines

SIGMADATA Learning

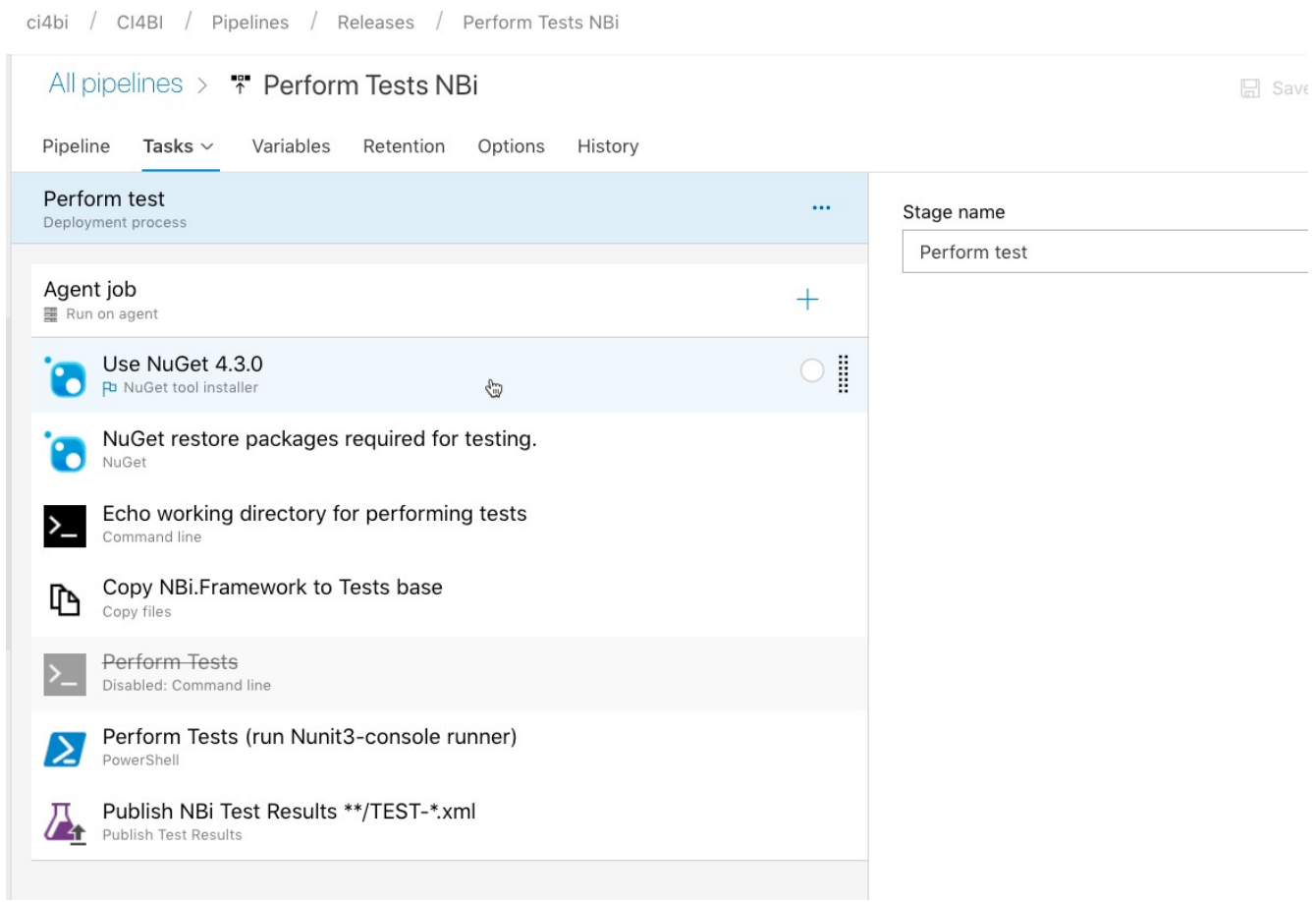
- Execute using NUnit console runner
  - Uses exit code as the number of failed tasks!
  - Azure Pipelines: *every exit code <> 0 is an error*
  - Select “continue on error”



- Integrate results using Publish Test Results task
- Supports JUnit, NUnit 2, NUnit 3, VSTest (TRX), xUnit 2

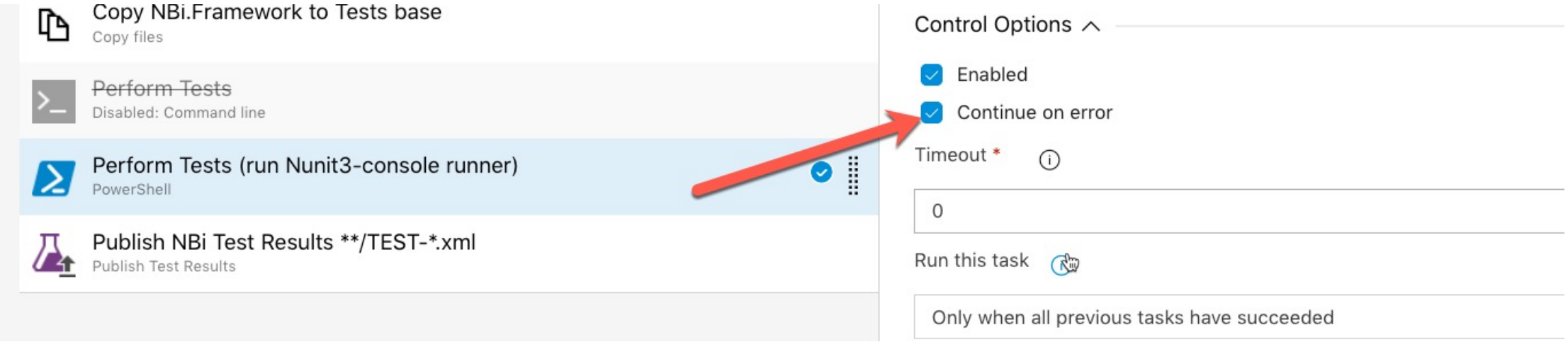


Perform NBi Test

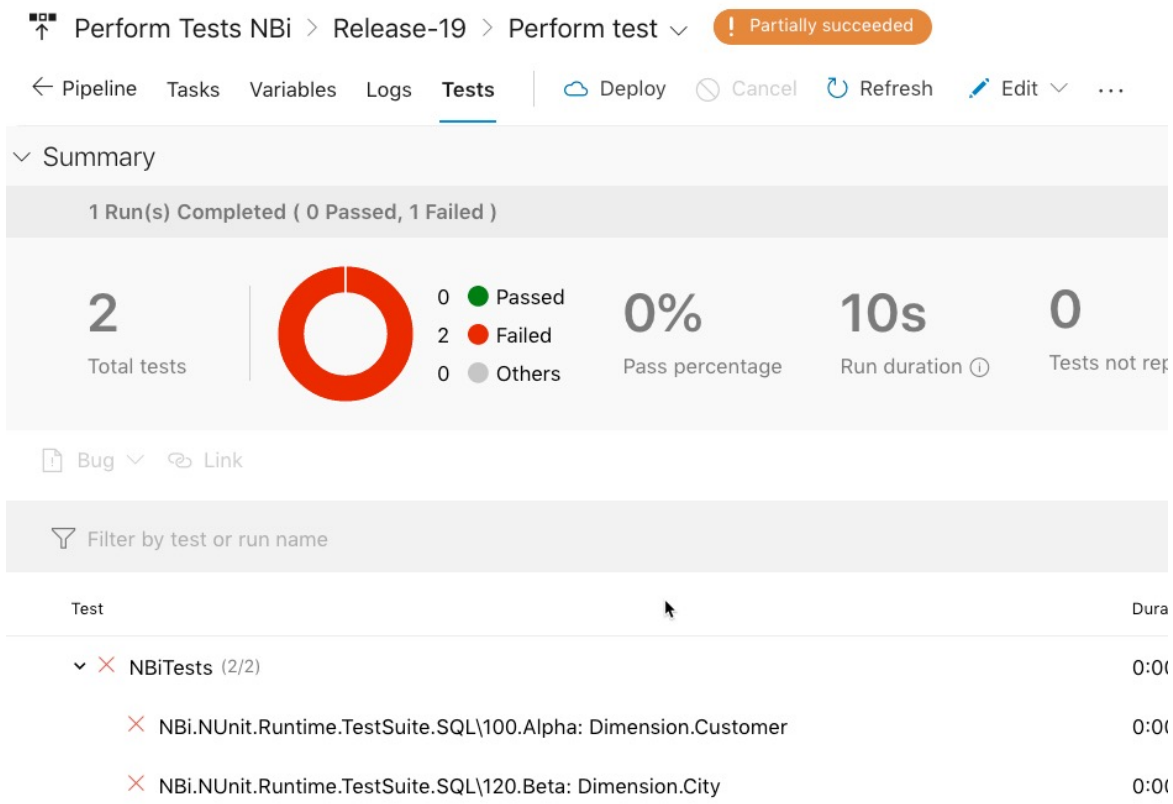


Pipeline Task will fail on test fail!

SIGMADATA Learning



Test Report Integration



Evaluatie

SIGMADATA
Learning

- http://bit.ly/devops-eval