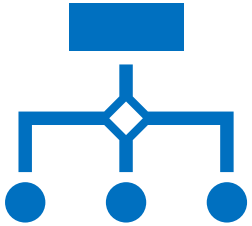
Unveiling the magic of CI/CD for SQL Server using GitHub Actions

Kevin Chant and Sander Stad

12/10/2022

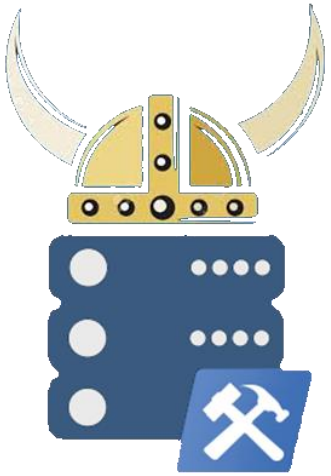
For the next 60 minutes



Sander Stad

DATAmasterminds

Microsoft® MVP Most Valuable Professional



@sqlstad

sqlstad.nl

sander@sqlstad.nl

github.com/sanderstad

Kevin Chant

- Lead BI & Analytics Architect in the Netherlands
- Worked in IT since Windows 95
- Experience in various sectors
- Various certifications, dual-category MVP

- Twitter: @kevchant
- LI: https://www.linkedin.com/in/kevin-chant/
- Blog: https://www.KevinRChant.com
- GitHub: https://github.com/kevchant

Branch strategies

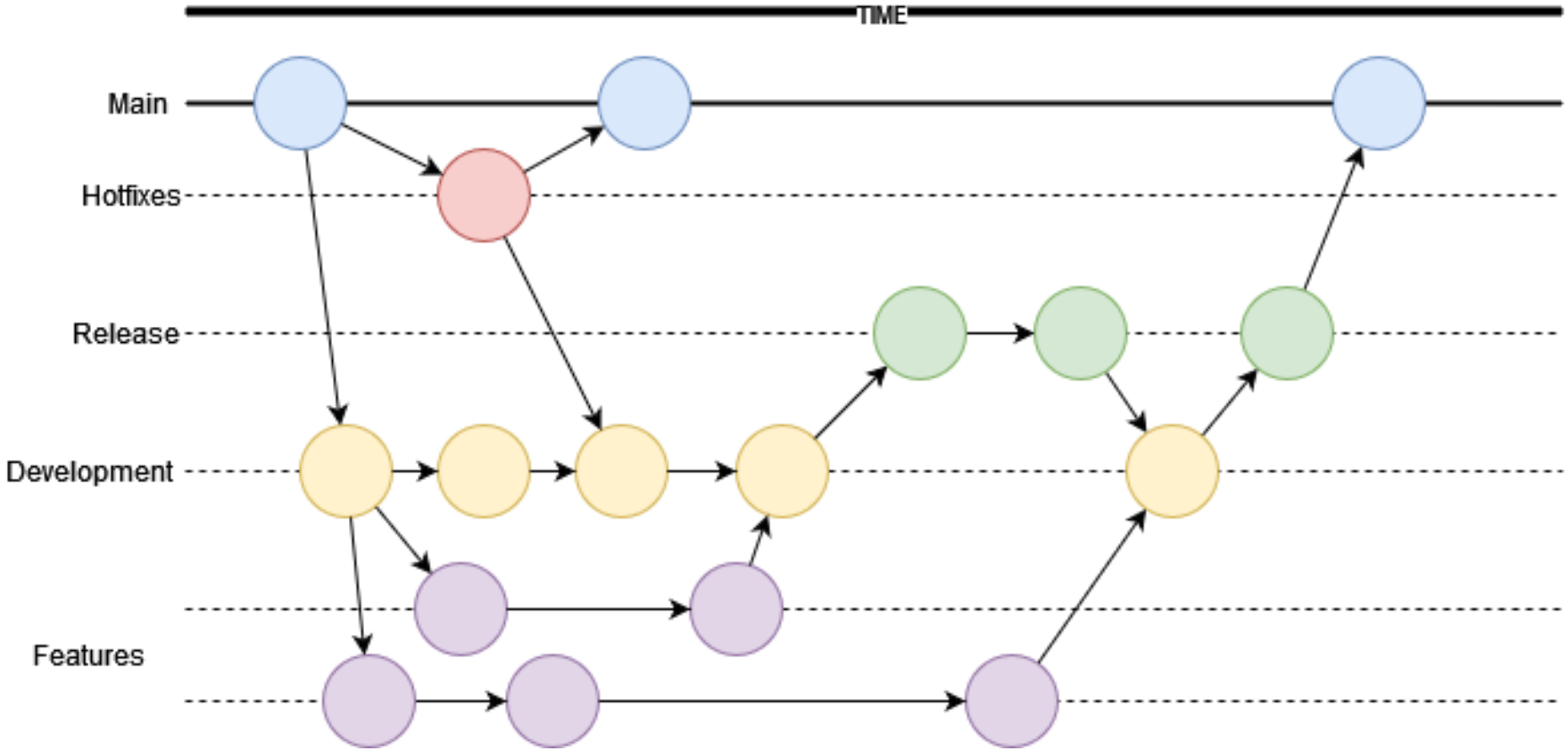
Branches

- What are they?

Examples of Branching models

- GitFlow
- Trunk-based Development (GitHub Flow, Microsoft Release Flow)
- Gitlab Flow

Git Flow example



Git Flow Pros and Cons

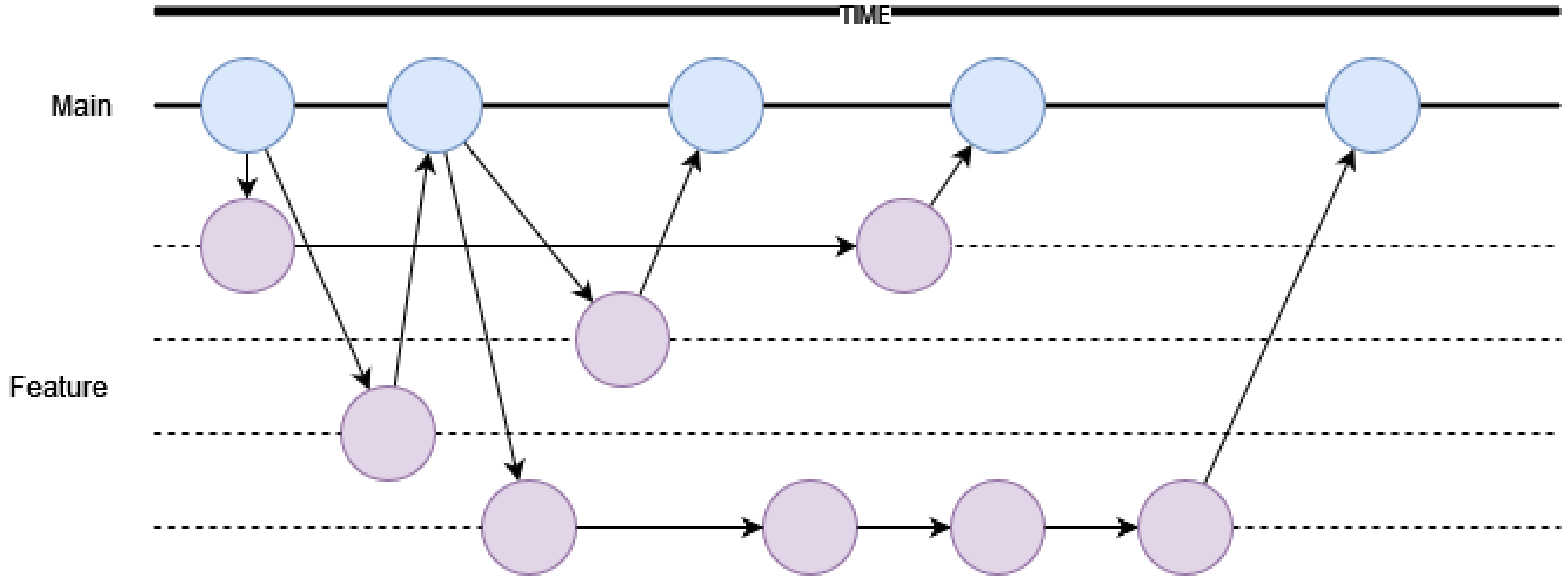
Pros

- Complete set of rules for all branches
- Very thorough and detailed version control
- Easy to scale because is simplifies parallel processing
- It’s easy to switch between development and other branches especially with CI/CD

Cons

- Complex and complicates because of many branches
- The methodology conflicts with the Agile methodology

GitHub Flow example



GitHub Flow Pros and Cons

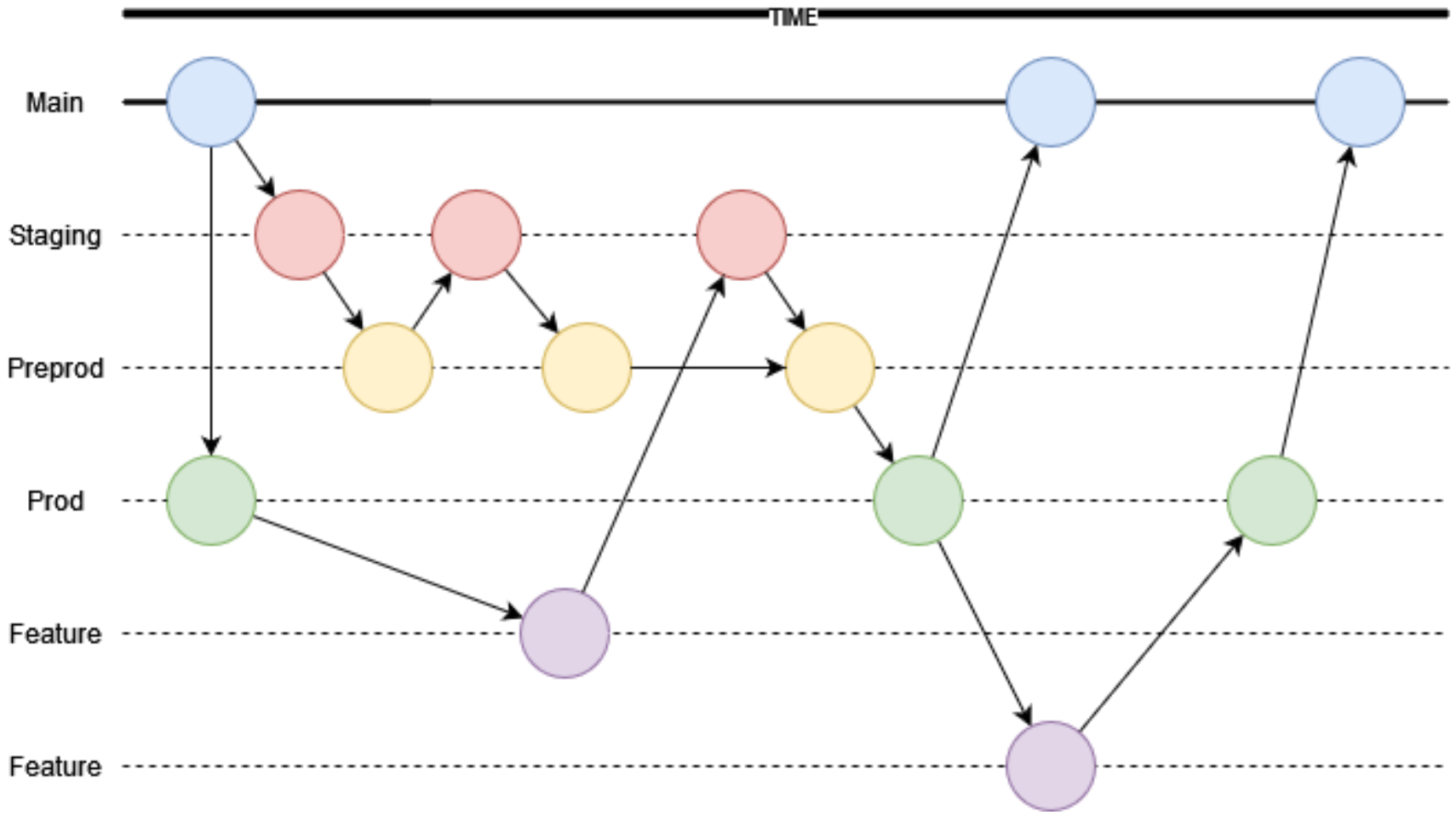
Pros

- Clear and simple collaboration
- Makes is easier for teams to quickly make changes
- Continuous deployment is a must. Features don’t stay in a branch until a release is made

Cons

- Speed comes at a cost and you see that it is not well organized
- May make it harder to manage the overall development process
- This may work well for software teams, but sometimes you need larger releases
- Harder to test multiple features together before deployment
- Main branch function both as the development and production branch

GitLab Flow example



GitLabFlow Pros and Cons

Pros

- Everything is tested in all environments
- You always know what lives in production
- There is only one way merges, downstream

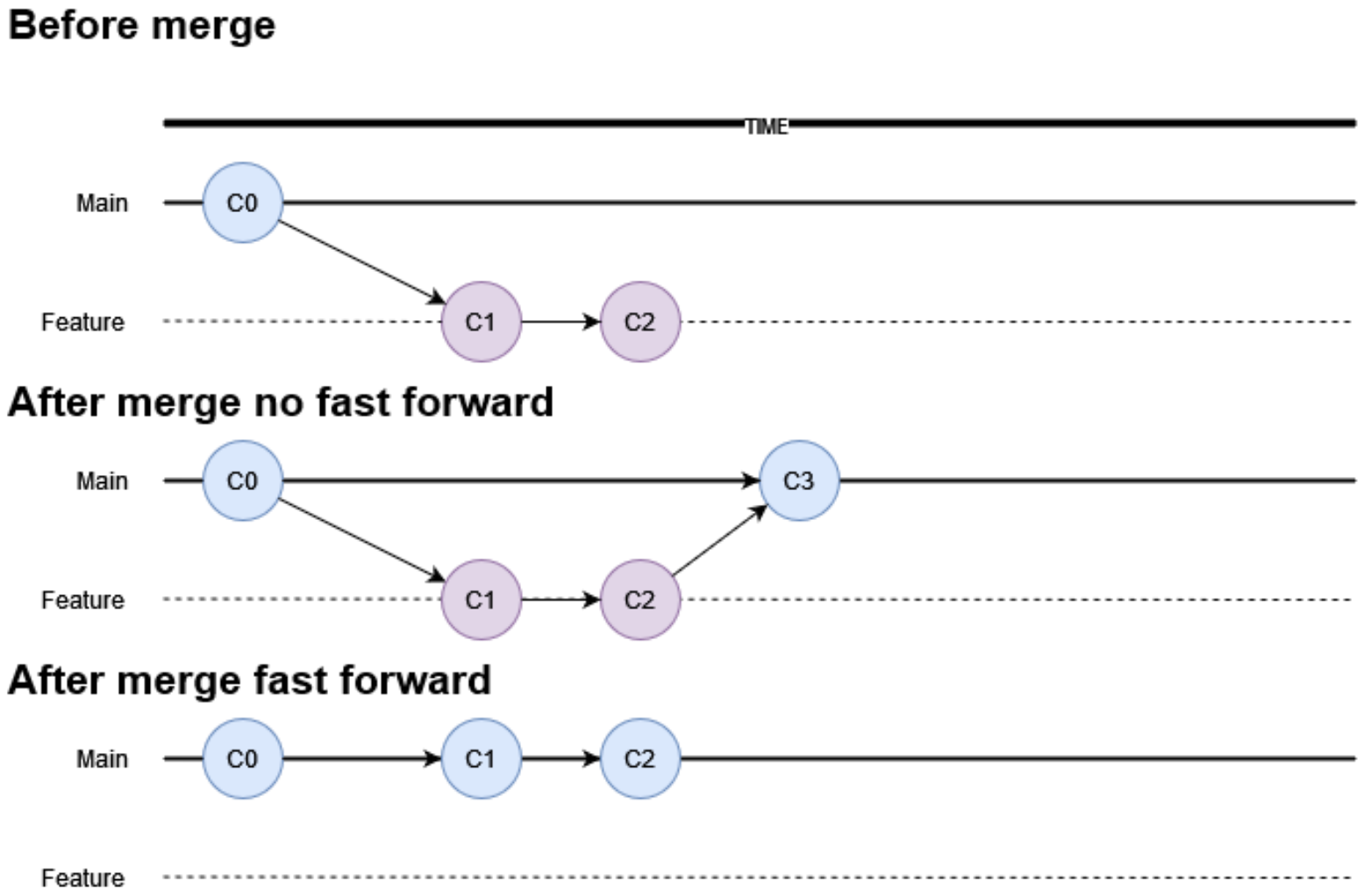
Cons

- You have to assume the main branch is free of errors
- There is no release validation to test
- Hotfixing production requires you to merge the feature branch with all the other branches as well

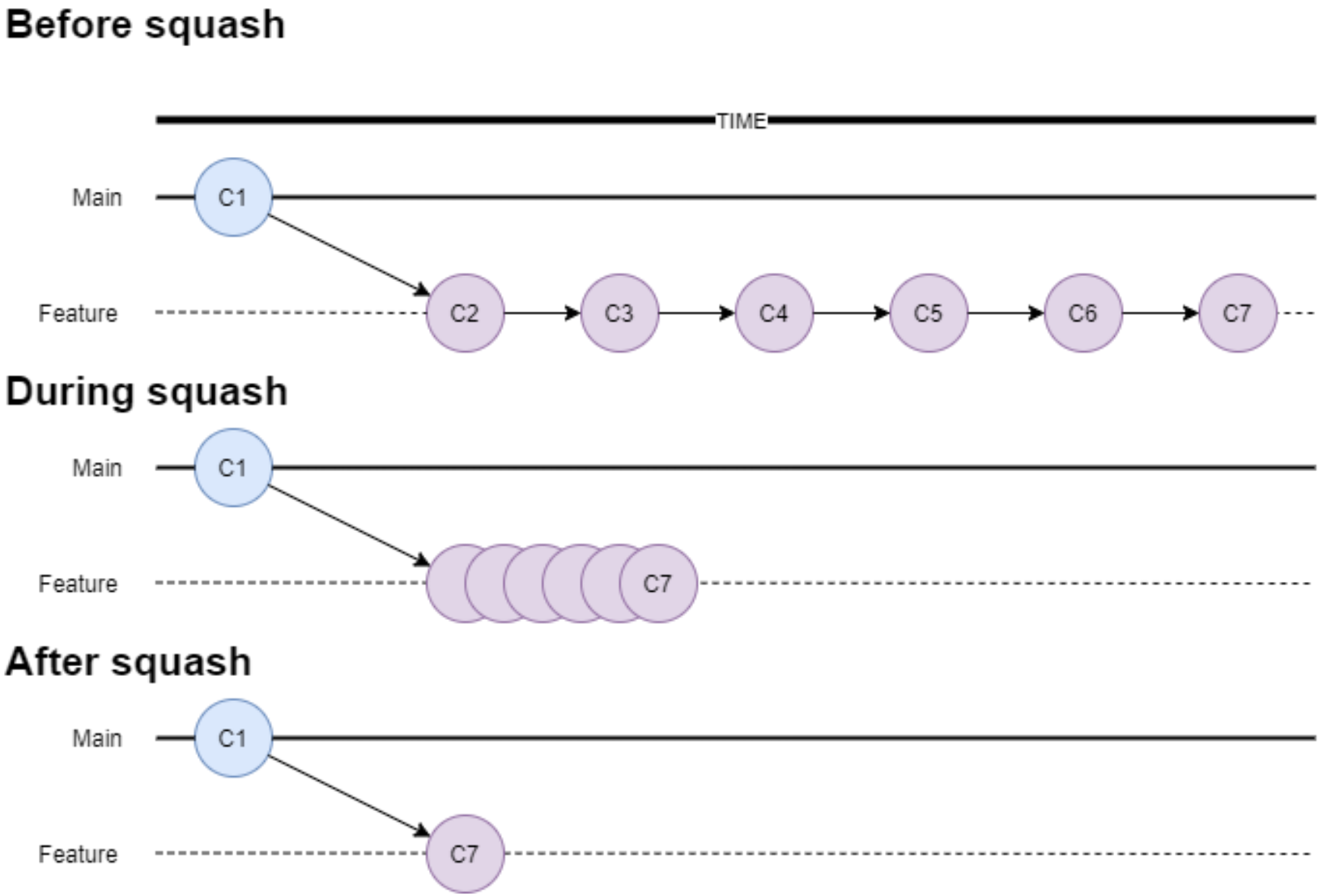
Popular merge strategies

- Merge with fast forward
- Merge with no fast forward
- Squash
- Rebase

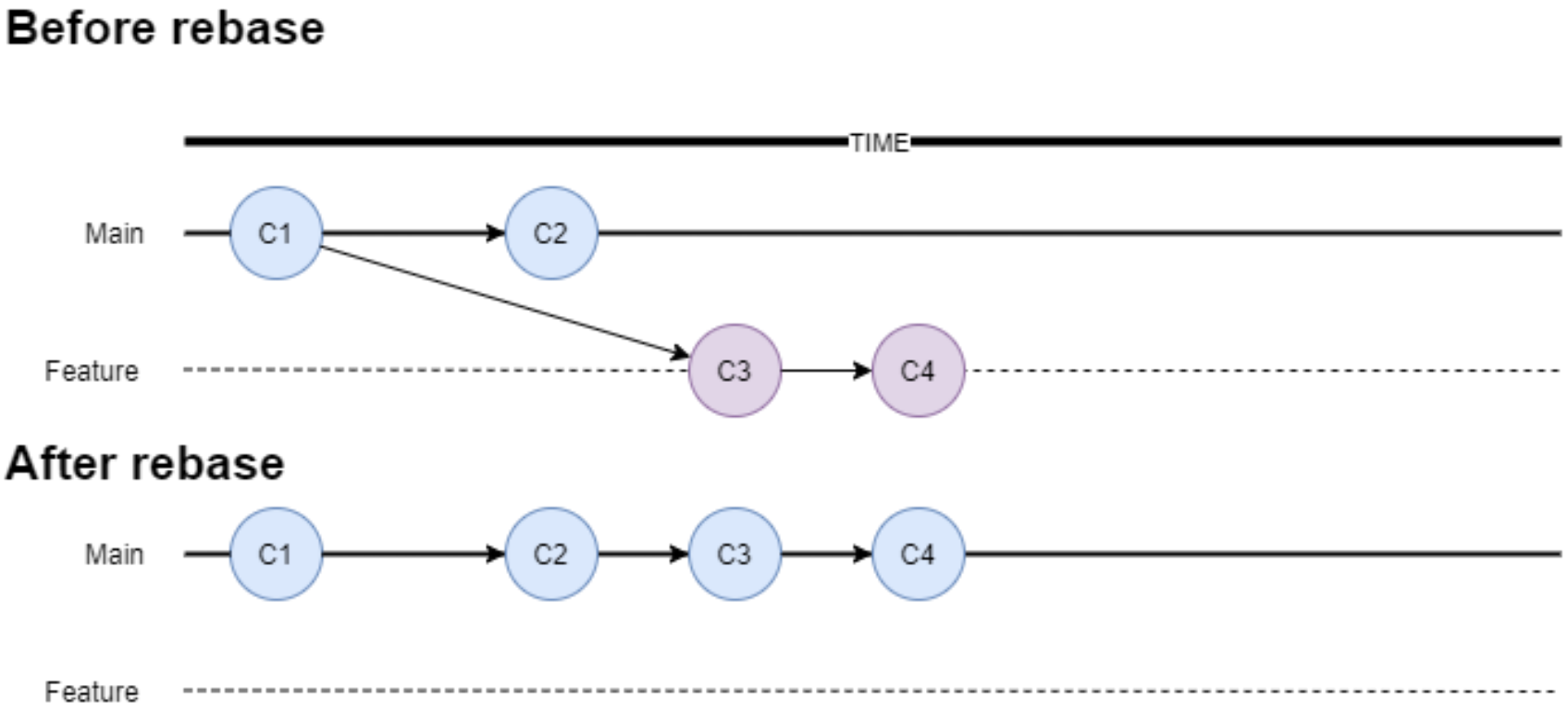
Merge strategies: Merge (no fast forward and fast forward)



Merge strategies: Squash



Merge strategies: Rebase



Common applications for Git

- Git SCM for Windows/Linux (required)
- Git GUI for Windows
- Visual Studio (& code)
- Azure Data Studio

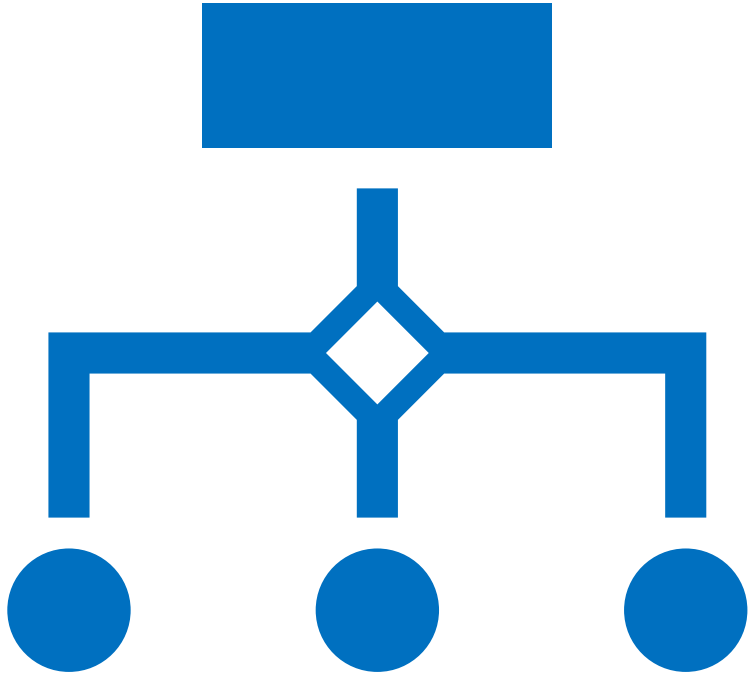
GitHub Runners

- Deal with all processing
- Runs on Windows, Ubuntu or MacOS
- GitHub or self-hosted
- GitHub-hosted image same as Azure Pipeline Agents
- Windows & Linux run on Standard_DS2_v2 images
- macOS images always run in US

Optimal settings

- GitHub hosted useful for cloud
- Self-hosted for local deployments
- Always self-hosted for custom apps
- Avoid running as service on laptop
- Create runner at right level

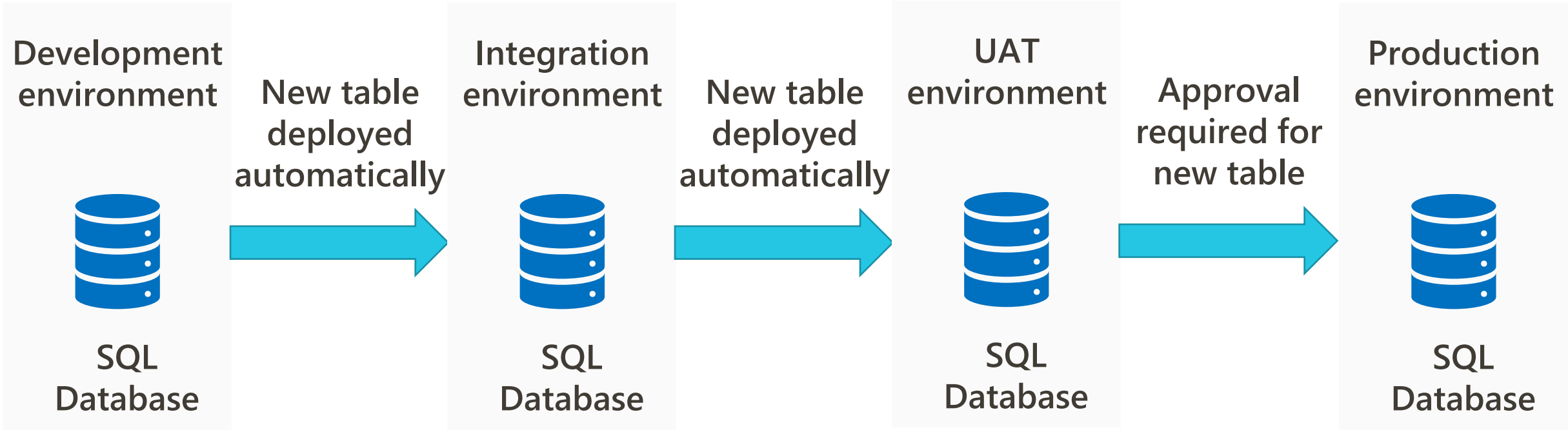
Configuring workflow for SQL Server deployments



Deploying SQL Server methods

- ARM templates
- Bicep
- Terraform
- Pulumi

SQL Server Database pipeline



Keep your secrets secret

Unit testing

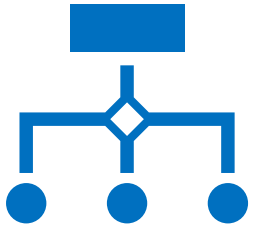
Magic

- Actually, GitHub repos
- Three types
- Deployments and automation
- Various types of triggers
- GitHub script
- Power BI

To recap








Questions?

ThankYou

Sander Stad	Kevin Chant
@sqlstad	@kevchant
Sqlstad.nl	KevinRChant.com
sanderstad	kevchant