



Lektionstillfälle 1

"Introduktion & Python Flask"

Utbildare: Robert Westin

NACKADEMIN

Lektionstillfällets mål och metod

Mål med lektionen:

- Läsa igenom kursmål & kursplan
- Sätta upp labbmiljö
- GitHub
- Python Flask

Lektionens arbetsmetod/er:

- Föreläsning
- Labb

Vem är ni? Frågor att svara på

- Vad är ditt namn? Repetition
- Vad är dina förväntningar på denna kurs?

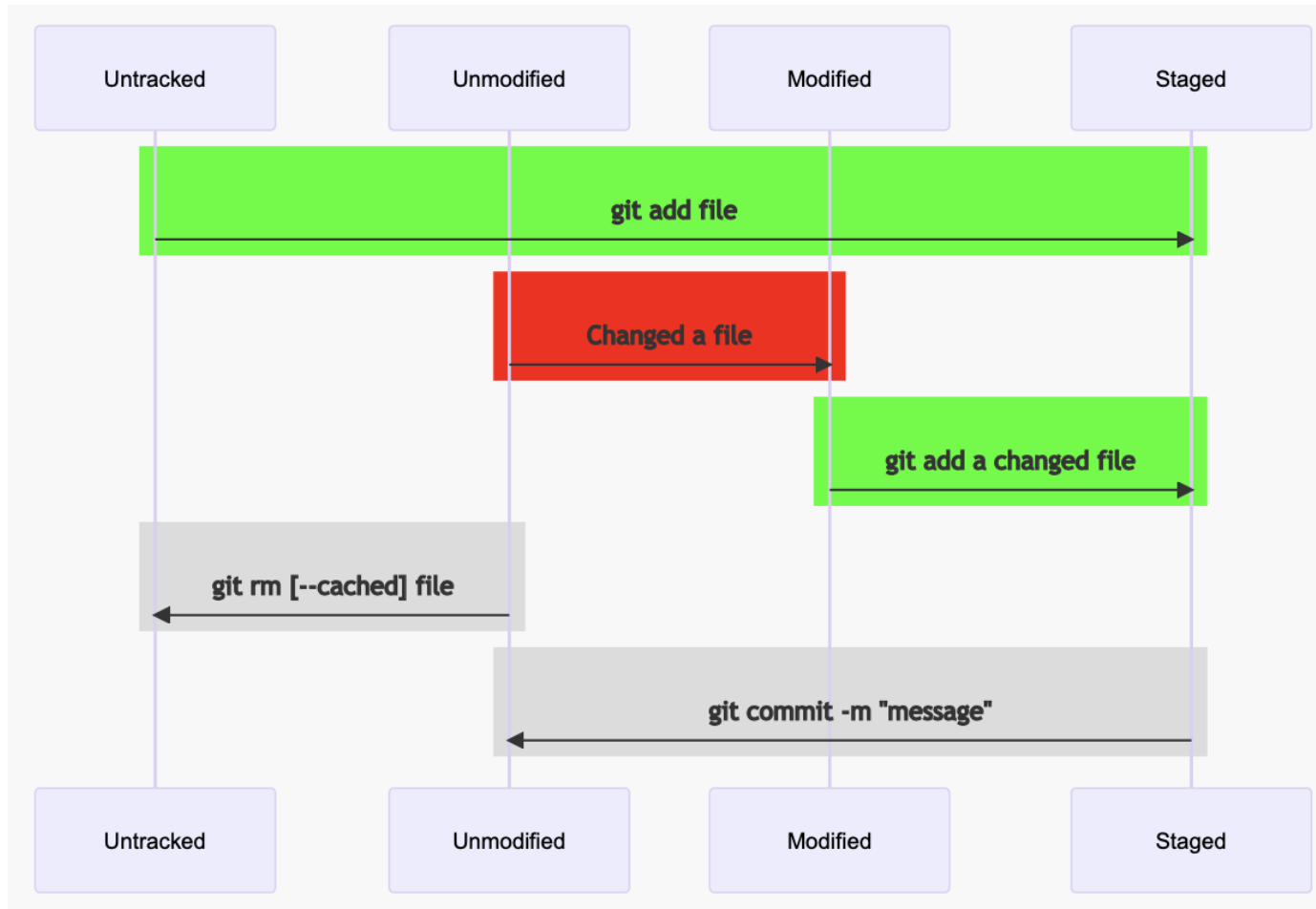
Begreppsgenomgång

- Continuous Integration
- Continuous Delivery & Continuous Deployment
- Python Flask
- GitHub Organization
- GitHub Repository
- GitHub Collaborator
- GitHub Teams
- GitHub Actions
- GitHub Classroom

Git

- Skapat av Linus Torvalds 2005 för utveckling av Linux kärnan
- Mål
 - Distribuerat
 - Snabbt
 - Data Integritet
 - Flera parallella utvecklingsspår
- Ingen central användarhantering

Git Grunder



NACKADEMIN

DVCS

- Distributed version control systems
- Vad innebär det att git är distribuerat?
 - Alla har varsin komplett version av sin kodbas
 - Fördelar?
 - Nackdelar?

Git Nackdelar

- Saknar central användarkontrol (Exempelvis GitHub löser detta)
- Om någon pushar:
 - En stor binär, så finns den kvar i historiken och måste laddas ner av alla
 - .exe-filer
 - Bilder
 - Ljud
 - Filmer
 - Kompilerade filer eller foldrar såsom .env, node_modules, coverage
 - Något hemligt, såsom ett lösenord (så finns den kvar i historiken)

Git Fördelar

- Lätt att ändra spår (branch, patch etc.)
- Litet och snabbt <https://git-scm.com/about/small-and-fast>
- "Automatisk backup" genom användarna
- Stödjer olika arbetssätt <https://git-scm.com/about/distributed>
- Data Integritet för historiken <https://git-scm.com/about/info-assurance>
- Gratis och öppen källkod

Git Repository

- GitHub
 - 2008
 - Uppköpt av Microsoft för 65 miljarder 2018
 - 90 miljoner aktiva användare (2022)
- GitLab
 - 2014
- Kringtjänster
 - Paket
 - Sidor
 - Actions

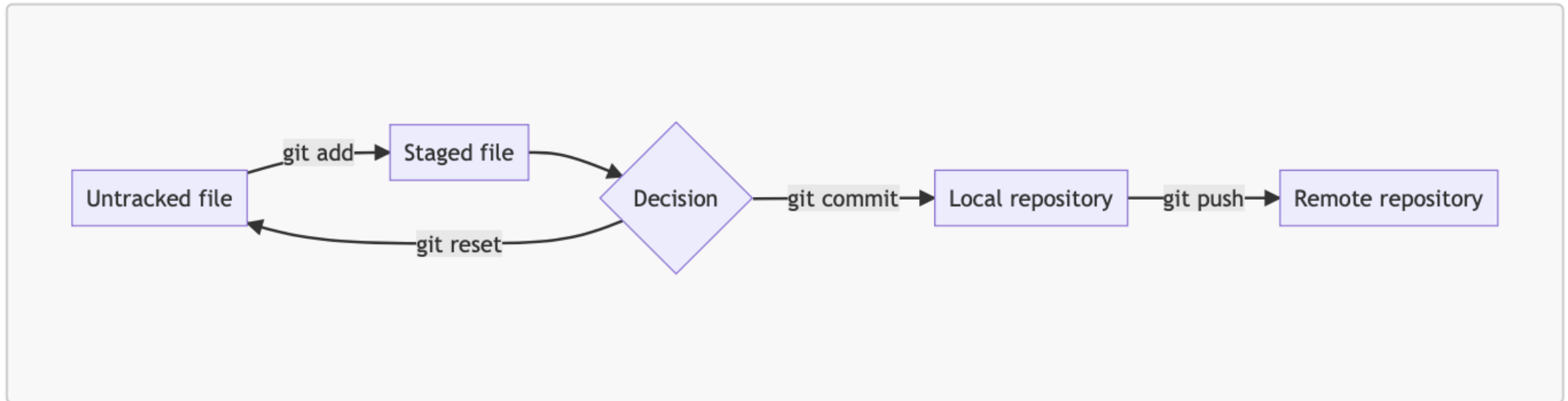
Git Repository

- Pull Request (GitHub) & Merge Request (GitLab)
- Användarhantering
- Integration med andra tjänster
 - T.ex kvalitetsverktyg
 - Deployment
 - Byggservrar med agenter
- Web IDE
- Protected Branches

Pull Requests

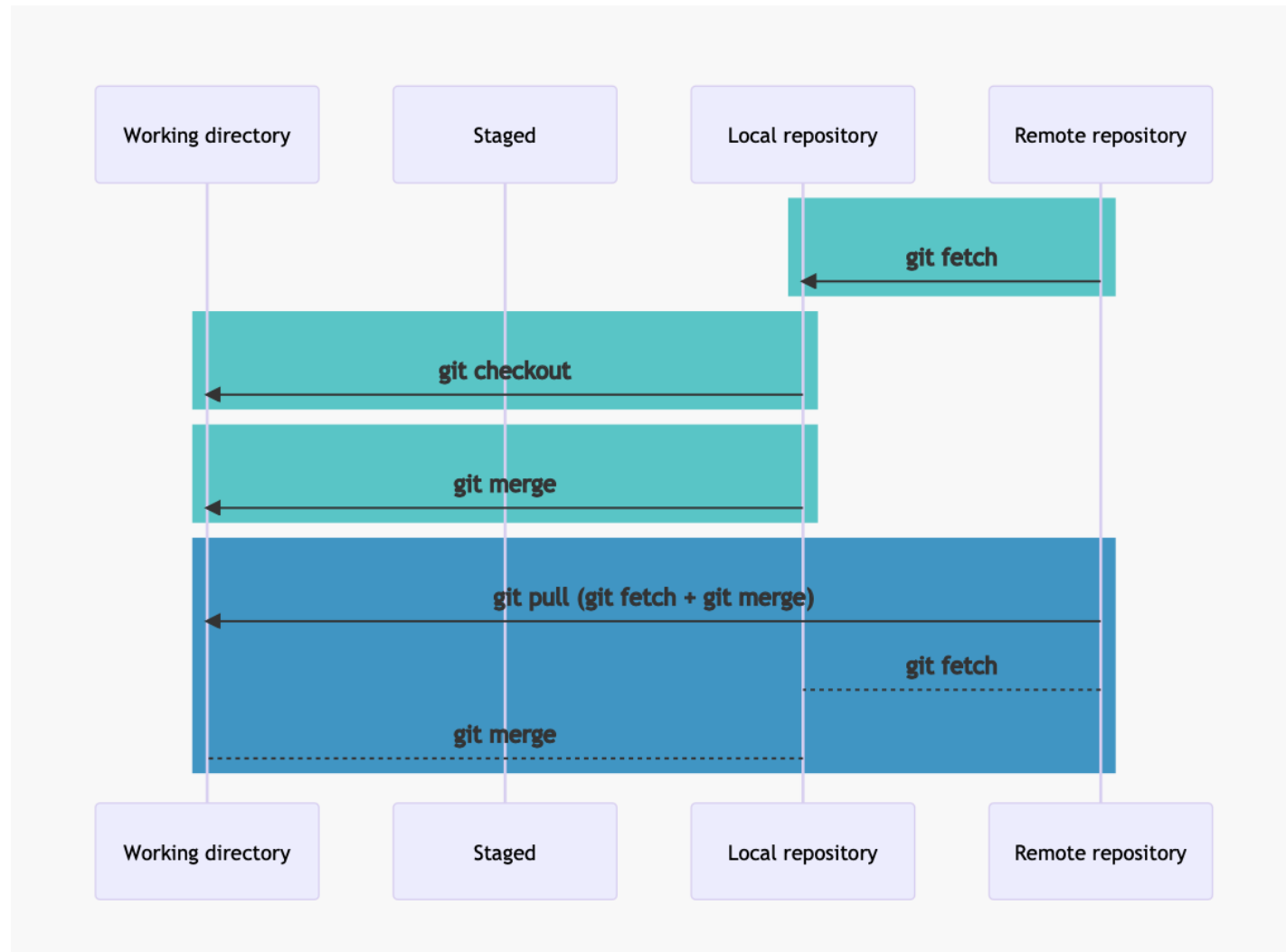
- Kommunikation och samarbetsverktyg för utvecklare
- Stödjer markdown & diagram
- Kan använda "Bot":ar för att kontrollera regler
- Kan kräva pipeline av tester är godkänd innan merge till main
- Kan kräva "Reviews" av flera utvecklare
- Kan kräva diskussioner är lösta

Git repetition - push



NACKADEMIN

Git repetition - fetch vs pull



Merge Conflict

- Vad är en merge konflikt?
 - Live demo
- Vems ändring är den rätta?
 - Varför har koden ändrats?
 - Är det något annat som inte kommer fungera om du ändrar?
- Vad händer om Pull Requesten innehåller dålig kod?
 - Koden kompilerar inte (inte så farligt)
 - Krashar i runtime (inte bra)

GitHub Merge konfiguration exempel

Protect matching branches

☐ **Require a pull request before merging**
When enabled, all commits must be made to a non-protected branch and submitted via a pull request before they can be merged into a branch that matches this rule.

☐ **Require status checks to pass before merging**
Choose which [status checks](#) must pass before branches can be merged into a branch that matches this rule. When enabled, commits must first be pushed to another branch, then merged or pushed directly to a branch that matches this rule after status checks have passed.

☐ **Require conversation resolution before merging**
When enabled, all conversations on code must be resolved before a pull request can be merged into a branch that matches this rule. [Learn more](#).

☐ **Require signed commits**
Commits pushed to matching branches must have verified signatures.

☐ **Require linear history**
Prevent merge commits from being pushed to matching branches.

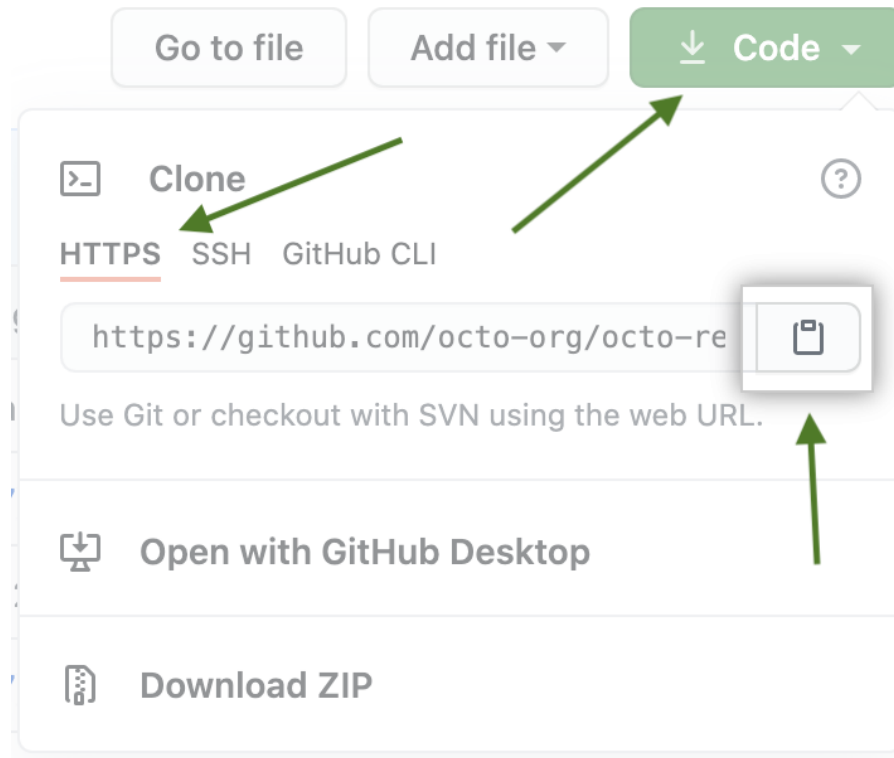
☐ **Require deployments to succeed before merging**
Choose which environments must be successfully deployed to before branches can be merged into a branch that matches this rule.

☐ **Include administrators**
Enforce all configured restrictions above for administrators.

☐ **Restrict who can push to matching branches**
Specify people, teams, or apps allowed to push to matching branches. Required status checks will still prevent these people, teams, and apps from merging if the checks fail.

NACKADEMIN

GitHub Grunder – Clona https



- <https://docs.github.com/en/repositories/creating-and-managing-repositories/cloning-a-repository>

NACKADEMIN

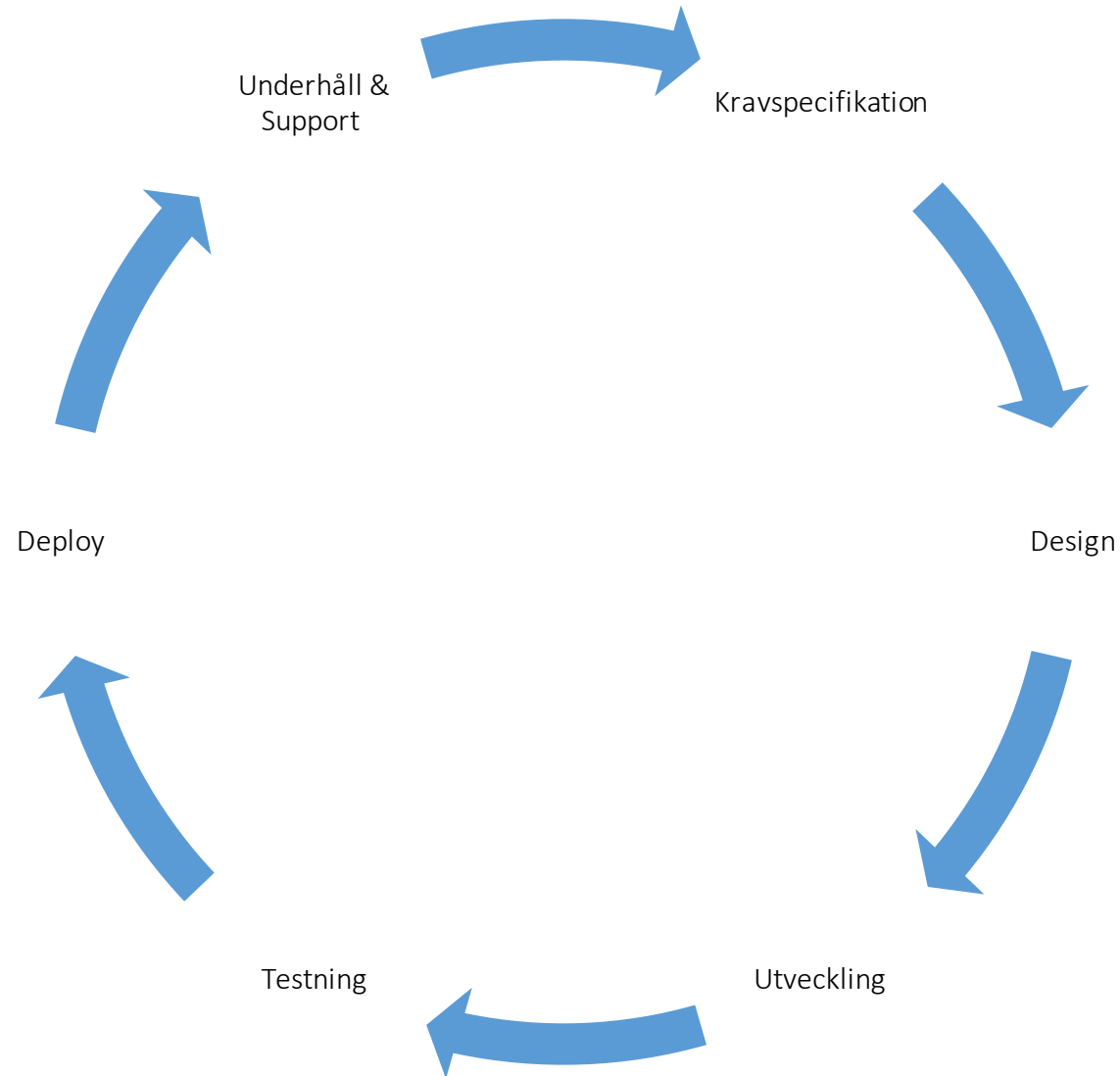
GitHub Grunder – SSH

1. Skapa ssh nyckel (om du inte redan har)
2. Lägg till publik nyckel i GitHub
3. Testa login i terminal: `ssh -T git@github.com`
4. Hämta url för ssh clone (se https clone, specifik flik för ssh)
5. `git clone <url>`

<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account>

NACKADEMIN

En Applikations livscykel



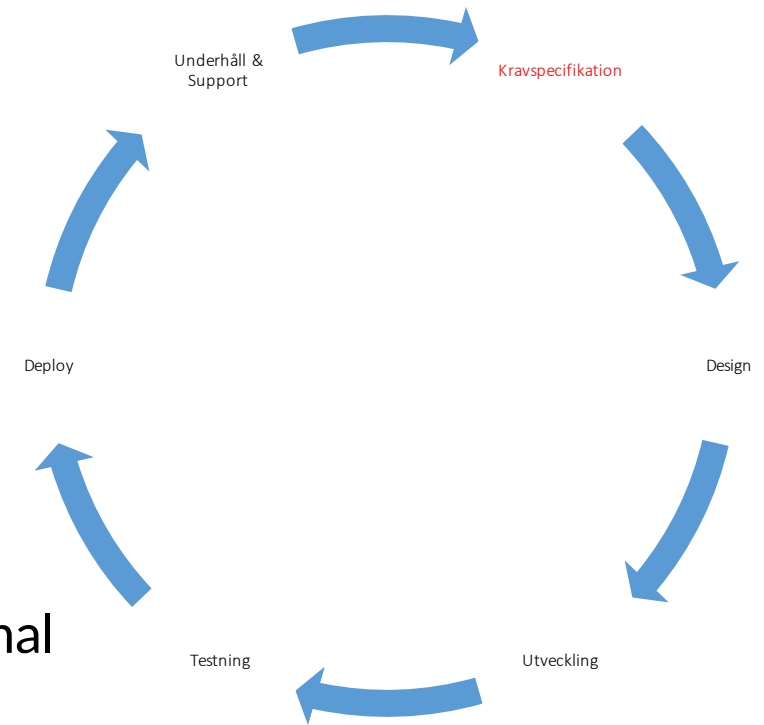
NACKADEMIN

Kravspecifikation

- Funktionella krav
 - Ett specifikt beteende
 - Det ska gå att skapa en användare som slutanvändare
 - Det ska gå att skapa en användare som support personal
 - Lösenordet ska sättas av slutkund vid första inloggning

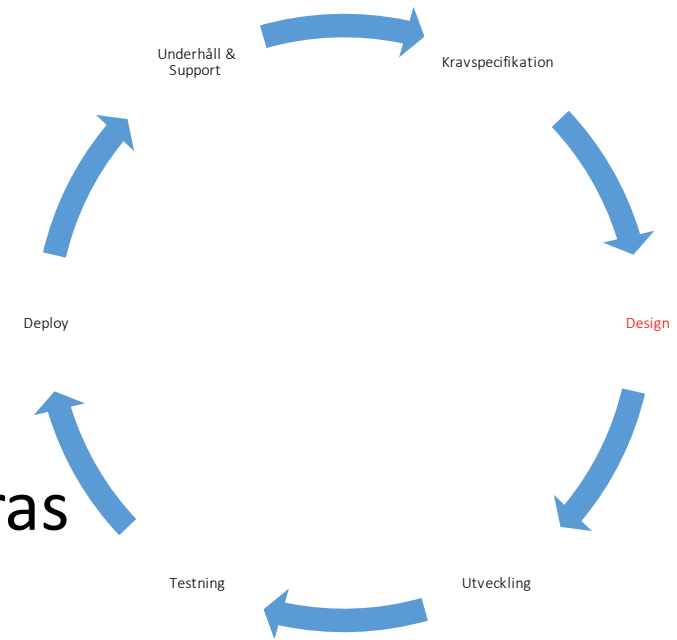
Icke-funktionella krav

- "Allt annat"
 - Säkerhet, svarstid, distribuering, observerbarhet, tålighet
- https://en.wikipedia.org/wiki/Non-functional_requirement



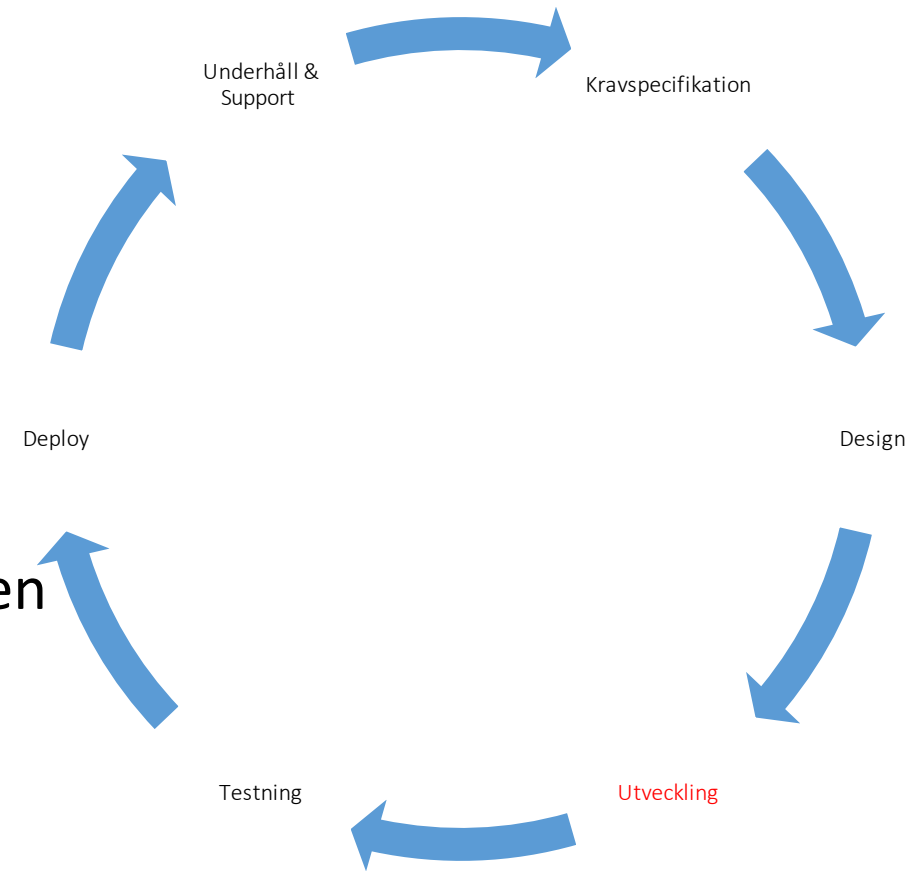
Arkitektur & Design

- Arkitektur löser frågor på systemnivå
- Design handlar ofta om hur koden ska implementeras
- Att hoppa över arkitektur & design är ett vanligt fel
- Att konstant tänka igenom sina lösningar ger arbetet högre kvalitet
- "Att tänka först och koda sedan"
 - Ger oftast en mer kvalitativ lösning på kortare tid.
- Att INTE designa sin kod resulterar i spaghettikod



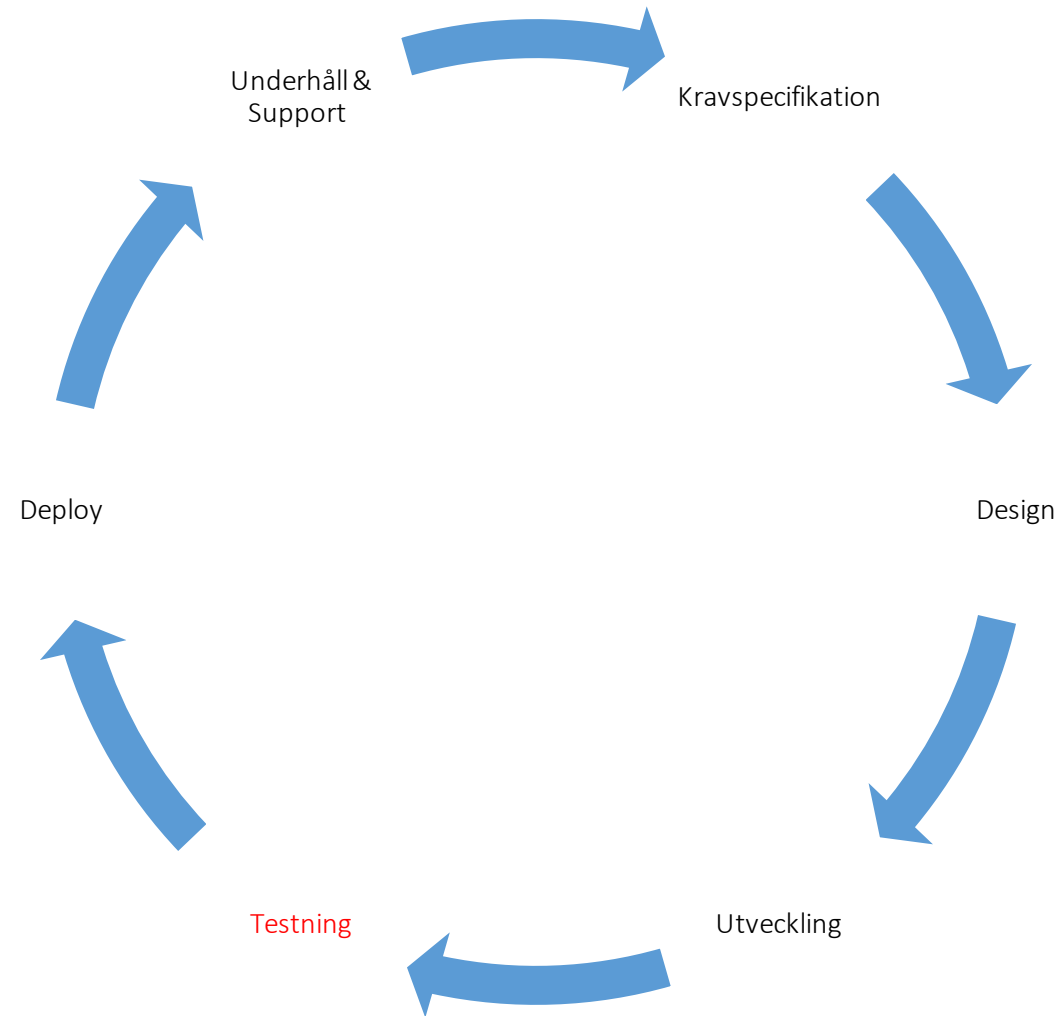
Utveckling

- Att koda enligt design
- Att koda enligt kraven
 - "Less is more" Skriv det som behövs för uppgiften
- Dockerfile
- Att testa sin kod
 - Testdriven utveckling (TDD)
 - RED – GREEN – REFACTOR



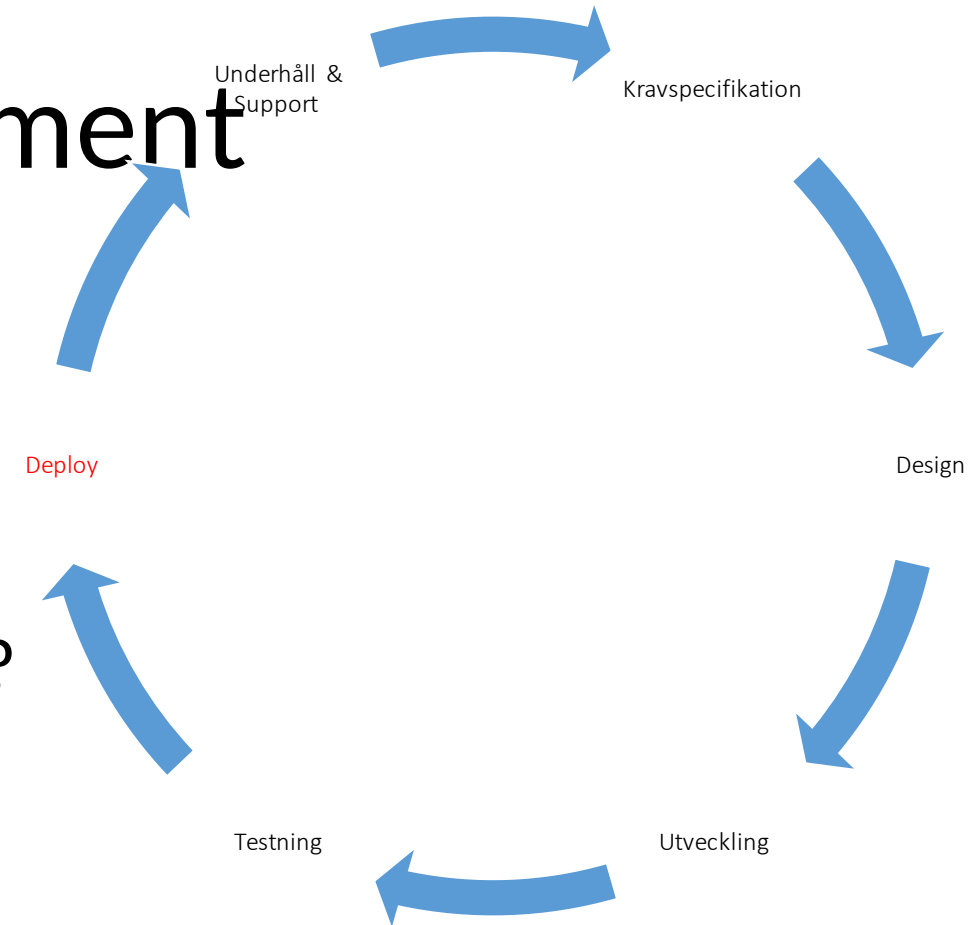
Test

- Inre och yttre loop, utvecklarens miljö och CI/CD miljön
 - Inre loop, koda, bygga, testa, commit
 - Yttre loop, Push till git -> CI -> CD?
- Enhetstester, Integrationstester, Prestandatest?, Säkerhet?



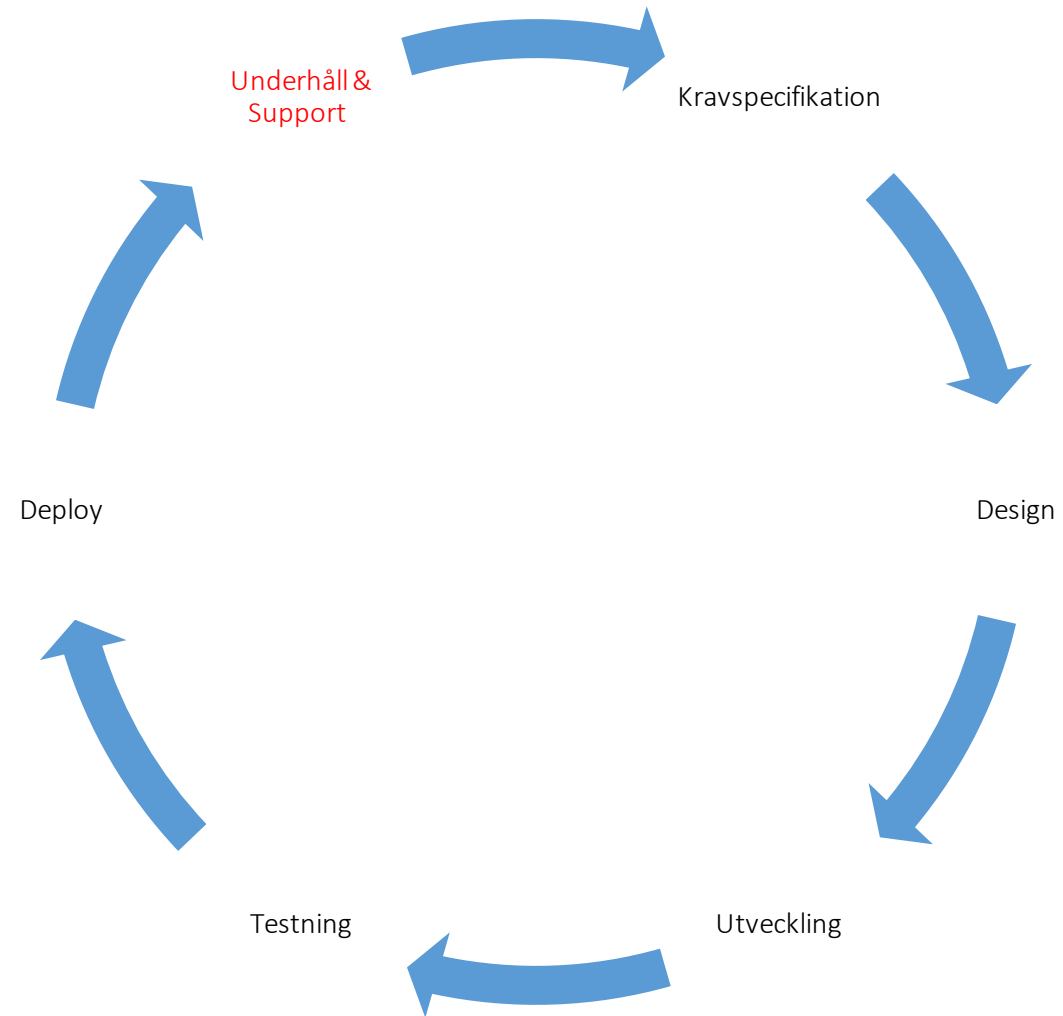
Deploy & Change Management

- Spårbarhet, loggar, audit trails etc.
- Varför görs ändringen?
- Vem gör ändringen?
- Vilka delar av systemet kan vi ändra själva?



Drift och förvaltning

- Vad ska vi göra om x, y, z händer?
- Hur viktigt är system A, B, C?
- Behöver vi ringa någon mitt i natten för att fixa problemet? Kan det vänta?
- Kan vi se historik vad som ändrat?
 - Är det något som vi kan backa?



Att bygga mjukvara

- Att koda är roligt och avgränsat i logikens värld
 - Men ibland kan ett "," orsaka flera timmar felsökning
 - Alla tänker inte lika när vi kodar
- Känner ni igen er? Berätta om något som ni lade orimligt lång tid på!
- När många ska jobba tillsammans, växer problemen
- Person 1 pushar kod, som kraschar för Person 2, vem fixar?
- Här är Continuous Integration "CI" en lösning!

CI - Continuous Integration


- Versionshantering av kod
- Automatisera "bygget"
 - Vad är ett kodbygge?
 - Vad är resultatet av ett bygge?
- Bygg varje ny kod ändring
- Automatisera tester
- Automatisera gemensamma regler & statisk kodanalys
 - Kodstil, komplexitet?
- Gick bygget bra eller dåligt? Resultat synligt!


Continuous Integration Tools



- Jenkins
- TeamCity
- Bamboo
- Travis CI
- GitLab CI
- Circle CI
- GitHub Action


GitLab Pipeline

Update .gitlab-ci.yml file









 4 jobs for [main](#)

 latest

 [ad828bee](#) 

 No related merge requests found.

Pipeline Needs Jobs 4 Tests 0

Build	Test	Deploy
<div> build-job </div>	<div> lint-test-job </div> <div> unit-test-job </div>	<div> deploy-job </div>

NACKADEMIN


CD - Continuous Delivery varför?



- Fokus på att alltid vara klar för release
- Nästan alla steg är automatiserade
- Maximera utvecklarnas tid för utveckling istället för administration
- Automatisera bort repetitiva uppgifter
- Ta bort den mänskliga faktorn
- Identifiera och ta bort saker som senarelägger "Time to market"
- Tillförlitliga leveranser med jämn kvalitet
- Undvik personberoenden, release Kim är sjuk idag?


NACKADEMIN

Continuous Delivery

Update .gitlab-ci.yml file

 6 jobs for [main](#) in 7 minutes and 1 second

 [85189977](#) 

 No related merge requests found.

Pipeline

Needs

Jobs 6

Tests 0



Build





Test



Test-deploy



Int-test



Deploy

 build-job 

 lint-test-job 
 unit-test-job 

 test-deploy... 

 int-test-job 

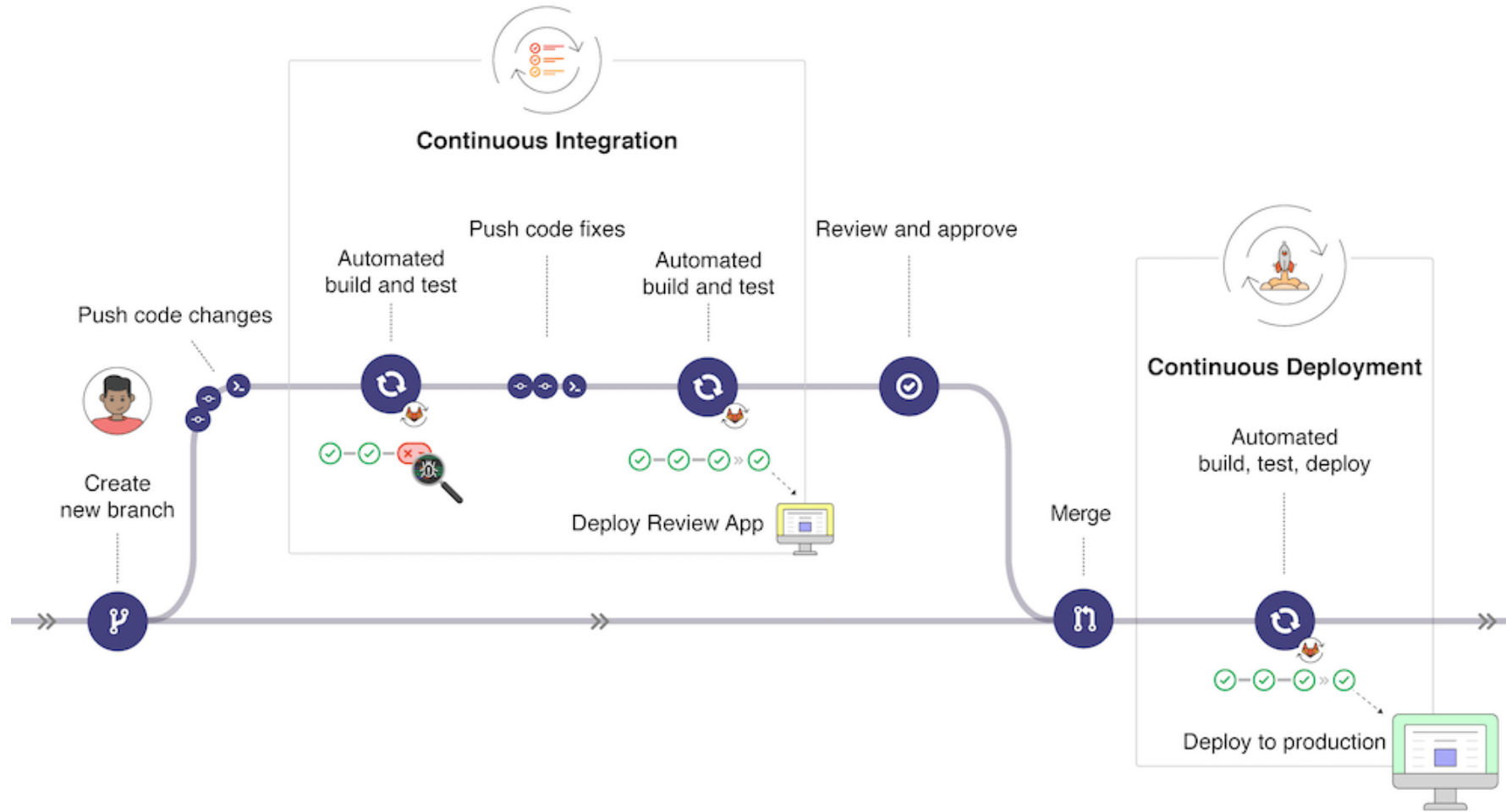
 deploy-job 

NACKADEMIN

CD - Continuous Deployment

- CD kan också betyda "Continuous Deployment"
- Deploy till produktion sker automatiskt
- Kräver Dev och Ops kompetens
- Hur vet vi om något går fel? Monitorering, alarm?
- Vad gör vi när något går riktigt fel?
 - disaster recovery plan
 - Rollback?
 - Backup?

Continuous Deployment

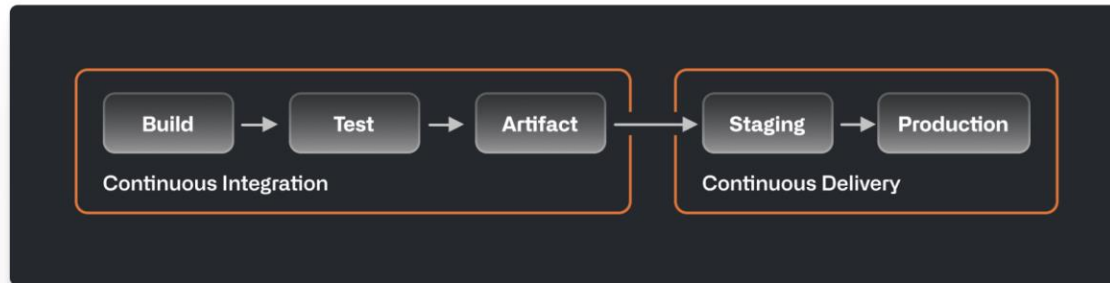


NACKADEMIN

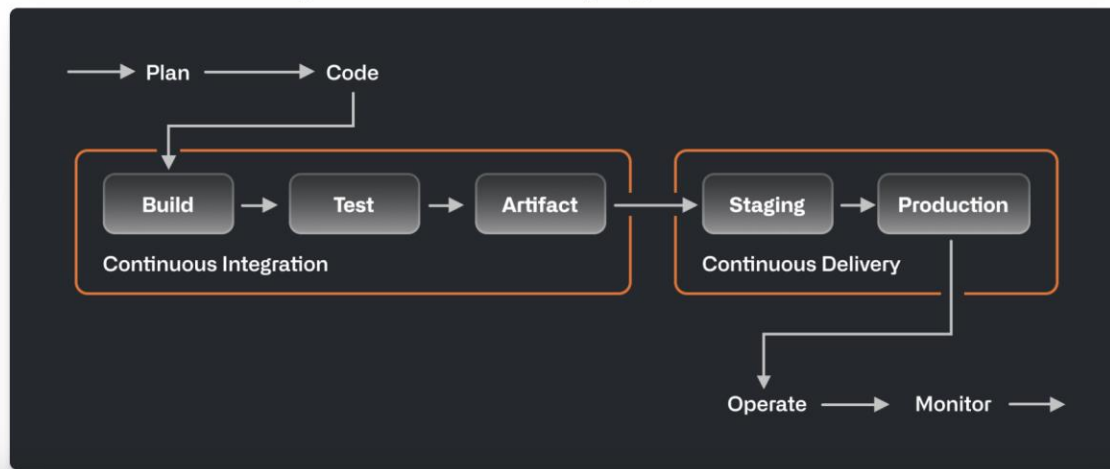
Källa: <https://docs.gitlab.com/ee/ci/introduction/>

GitHub CI/CD

A CI/CD pipeline



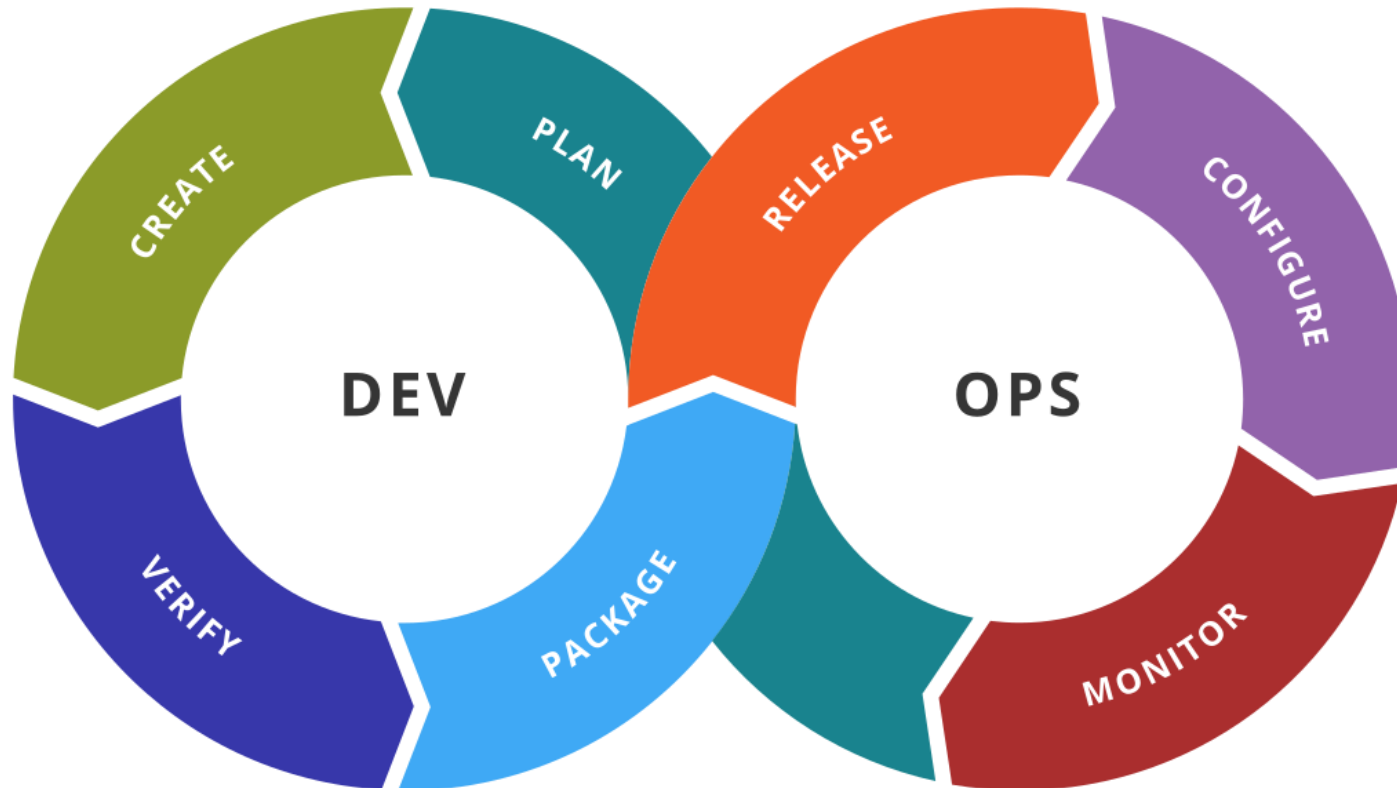
Continuous delivery vs. continuous deployment



NACKADEMIN

Källa: <https://resources.github.com/ci-cd/>

DevOps – Vart är CI? CD?



Källa: Wikimedia - Kharnagy

NACKADEMIN

Python Flask

Vi live kodar med Python Flask

NACKADEMIN

Laborationer

<https://classroom.github.com/a/vfQqXiwf>

NACKADEMIN

Summering av dagens lektion

- Vilka är vi?
- Git
 GitHub
- CI & CD
- Labbmiljö

Nästa lektion

- Docker
 - Grundläggande kommandon
 - Grundläggande funktionalitet
 - Nätverk
 - Felsökning
 - Tips & Tricks