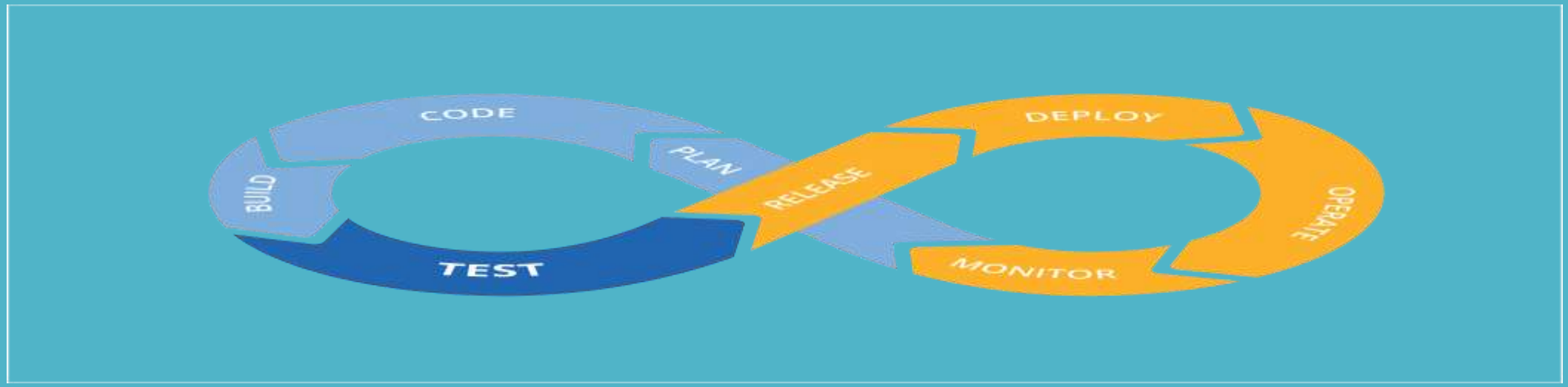
DevOps

(osnove)



O predmetu

Predmet je koncipiran da se prođu osnovne stari o Devopsu.

Kako je to veoma široka oblast mi ćemo se fokusirati na sledeće stvari, odnosno alate

* Git
*Jenskins
*Docker
*Kibernetes
*Azure Devops

Šta je DevOps

DevOps je pristup radu koji naglašava brzu, postepenu i kontinuiranu isporuku proizvoda.

Termin DevOps kombinuje reči „razvoj“ i „operacije“. U praksi, to je unija između razvojnih i operativnih timova.

DevOps je metodologija razvoja softvera koja se često smatra procesom, kulturom ili skupom principa koji omogućavaju organizacijama da brzo i kontinuirano isporučuju proizvode.

Ključna svrha DevOps-a

DevOps je kreiran kao odgovor na probleme koji su proizašli iz dugogodišnje tradicije na radnom mestu posedovanja izolovanih timova—ili potpuno odvojenih timova za razvoj, testiranje i operacije u vezi sa bilo kojim pojedinačnim proizvodom.

Ova izolovana struktura nije uvek pogodna za efikasnost, jer svaki tim ima sopstvene skupove prioriteta, zadataka i vremenskih rokova koji nisu nužno usklađeni sa okolnim timovima.

Ključna svrha DevOps-a je stvaranje kohezivnijeg razvojnog ciklusa.

Dodatne prednosti DevOps kulture uključuju poboljšanu efikasnost tima, povećanu brzinu objavljivanja i bolje mehanizme povratnih informacija.

DevOps metodologije naspram tradicionalnog procesa razvoja softvera

U kompaniji sa tradicionalnim procesom, programeri softvera bi napisali kod proizvoda, zatim bi ga predali timu za testiranje da testira funkcionalnost proizvoda, a zatim bi ga predali operativnom timu da dugoročno održava softver.

Ova izolovana struktura nije uvek pogodna za efikasnost, jer svaki tim ima sopstvene skupove prioriteta, zadataka i vremenskih rokova koji nisu nužno usklađeni sa okolnim timovima.

Sa DevOps pristupom, ti višestruki timovi su integrisani u jedan tim. Testiranje se može odvijati automatski i često tokom procesa uporedo sa razvojem proizvoda, a sve grupe mogu biti uključene u dugoročno održavanje.

Životni ciklus DevOps

Životni ciklus DevOps-a je integrativniji od procesa isporuke softvera u izolaciji. Primena proizvoda i ažuriranja se dešavaju neprekidno.

DevOps cevovod (pipelin) je manje rigidan, linearan proces.

Pošto rade kao jedinica, svaki član tima bi trebalo da bude ugodan u svakoj fazi životnog ciklusa, od početne ideje do procene kvaliteta softvera i razumevanja korisničkog iskustva.

Tokom procesa razvoja, DevOps timovi, uključujući programere softvera, analitičare za osiguranje kvaliteta (KA) i menadžere proizvoda, rade kao jedinica kroz faze ***planiranja, razvoja, isporuke i nadgledanja***.

Evo pregleda DevOps toka posla:

- U fazi ***planiranja***, tim otkriva probleme koje žele da reše i kako bi mogli da ih reše.
- Zatim će **razviti** svoj proizvod, koristeći okruženje za testiranje ili proizvodnju – bilo simulirano okruženje ili uzorkovanje korisnika iz stvarnog sveta da isprobaju ažuriranja pre nego što budu široko primenjena – da bi napravili najbolji mogući proizvod.
- Zatim će **isporučiti** proizvod svojoj široj publici.
- Konačno, **kontinuirano praćenje** osigurava stabilnost performansi.

Povratne informacije su ugrađene u kasnije iteracije i ažuriranja proizvoda—što vraća proizvode u fazu planiranja.

DevOps principi

Postoji nekoliko osnovnih principa na delu u DevOps inicijativi. U velikoj meri raščlanjeni, oni uključuju:

Sistemsko razmišljanje: Sistemsko razmišljanje znači razmišljanje o performansama celog sistema, umesto o performansama određenih timova. Ovakav način razmišljanja osigurava da se svi timovi i zaposleni osećaju odgovornim za proizvodnju dobrog kvaliteta i obeshrabruje timove da propuste nedostatke nizvodno.

Kultura: Uspešna DevOps kultura je često vezana za duh poboljšane saradnje, eksperimentisanja i kontinuiranog učenja. To bi moglo značiti da timovi osiguravaju da je vreme dodeljeno za poboljšanje rada, timovi su nagrađeni za preuzimanje rizika, a članovi mogu da uče od drugih unutar i van svojih timova.

Automatizacija: DevOps stavlja veliki naglasak na automatizaciju što je više moguće. Ovo može smanjiti vreme utrošeno na zadatke koji se ponavljaju i oduzimaju mnogo vremena i povećati brzinu primene. DevOps tim bi, na primer, mogao da automatizuje procese testiranja kako bi programeri mogli da dobijaju povratne informacije rano i često.

Upravljanje konfiguracijom: Upravljanje konfiguracijom je proces sistemskog inženjeringa. Fokusira se na uspostavljanje konzistentnosti proizvoda i održavanje tog nivoa efikasnosti tokom životnog ciklusa proizvoda.

DevOps prakse

Nekoliko ključnih praksi čine DevOps onim što jeste. Ovi uključuju:

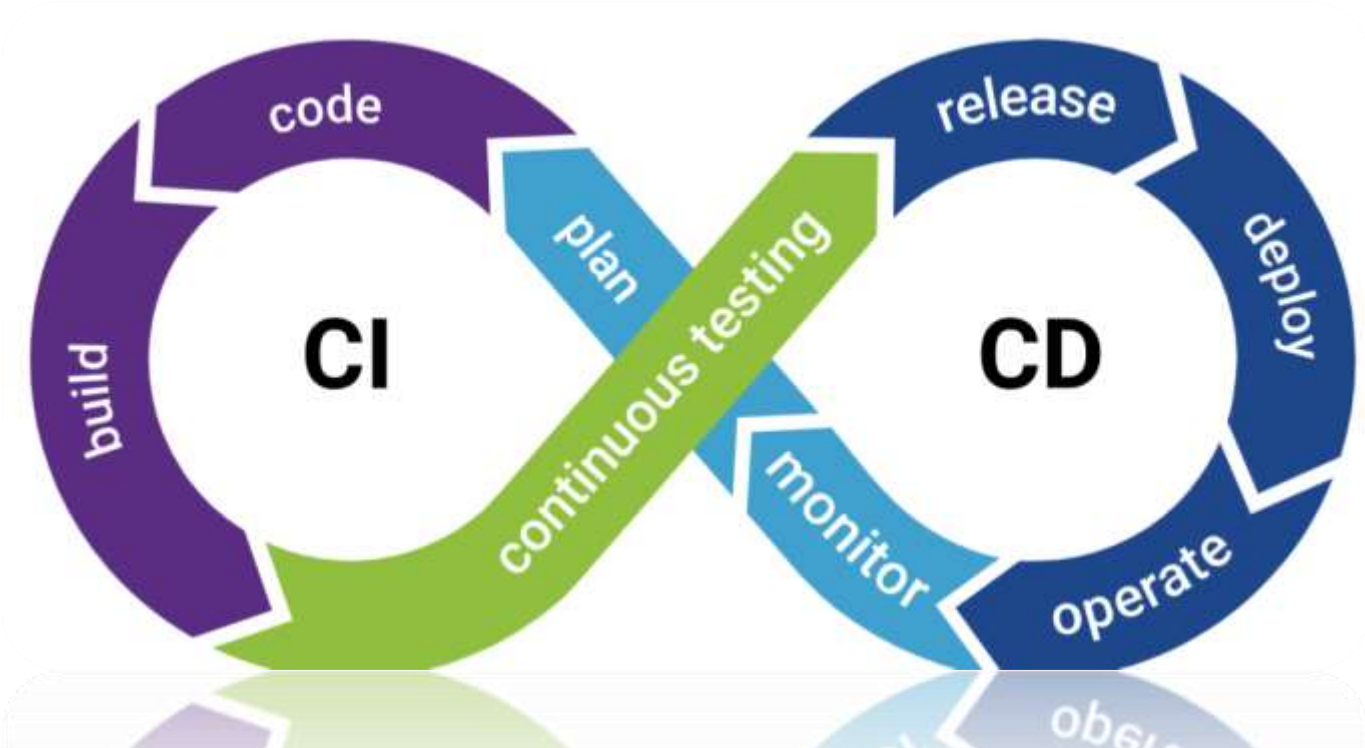
- **Kontinuirano testiranje**: Kontinuirano testiranje je proces koji ima za cilj povećanje efikasnosti razvoja softvera testiranjem i ugradnjom automatizovanih povratnih informacija tokom životnog ciklusa razvoja softvera (SDLC). Primena svih promena koda u okruženje za testiranje nakon faze izrade pomaže u praćenju grešaka pre nego što izazovu veće probleme.

- **Kontinuirana integracija (CI):** Kontinuirana integracija znači da su povratne informacije od zainteresovanih strana i popravke stalno integrisane u proizvod. To može značiti i automatizaciju procesa u koje su popravke integrisane, i stvaranje kulture u kojoj se dešava kontinuirana integracija.

- **Kontinuirana isporuka (CD):** Kontinuirana isporuka je kada se promene na proizvodu (verovatni kod) automatski integrišu. Ovaj metod brze isporuke osigurava da je proizvod uvek u stanju za postavljanje. To znači da se kod može primeniti u kratkim vremenskim okvirima (dnevno, nedeljno i tako dalje).

Zajedno, kontinuirana integracija i kontinuirana isporuka se često nazivaju **CI/CD.** Uzimajući ove prakse korak dalje, kontinuirana primena dodaje rutinu praćenja, testiranja i ažuriranja proizvoda u realnom vremenu nakon njihovog pokretanja.

U okviru DevOps okruženja, uobičajeno je da organizacije objavljuju manja, češća ažuriranja proizvoda koja reaguju na povratne informacije kupaca, a ne velika, radno intenzivna ažuriranja koja timovi mogu da primenjuju.



DevOps alati

Iako se DevOps prvo smatra načinom razmišljanja, postoji nekoliko alata za automatizaciju koji mogu pomoći u povećanju efikasnosti DevOps procesa:

Git: Git je sistem kontrole verzija. U DevOps-u se koristi za praćenje koda i korisno je za članove tima da sarađuju na projektima i ažuriraju postojeće.

Docker: Docker se koristi za kontejnerizovanje aplikacija—proces pretvaranja aplikacije u jedan paket softvera.

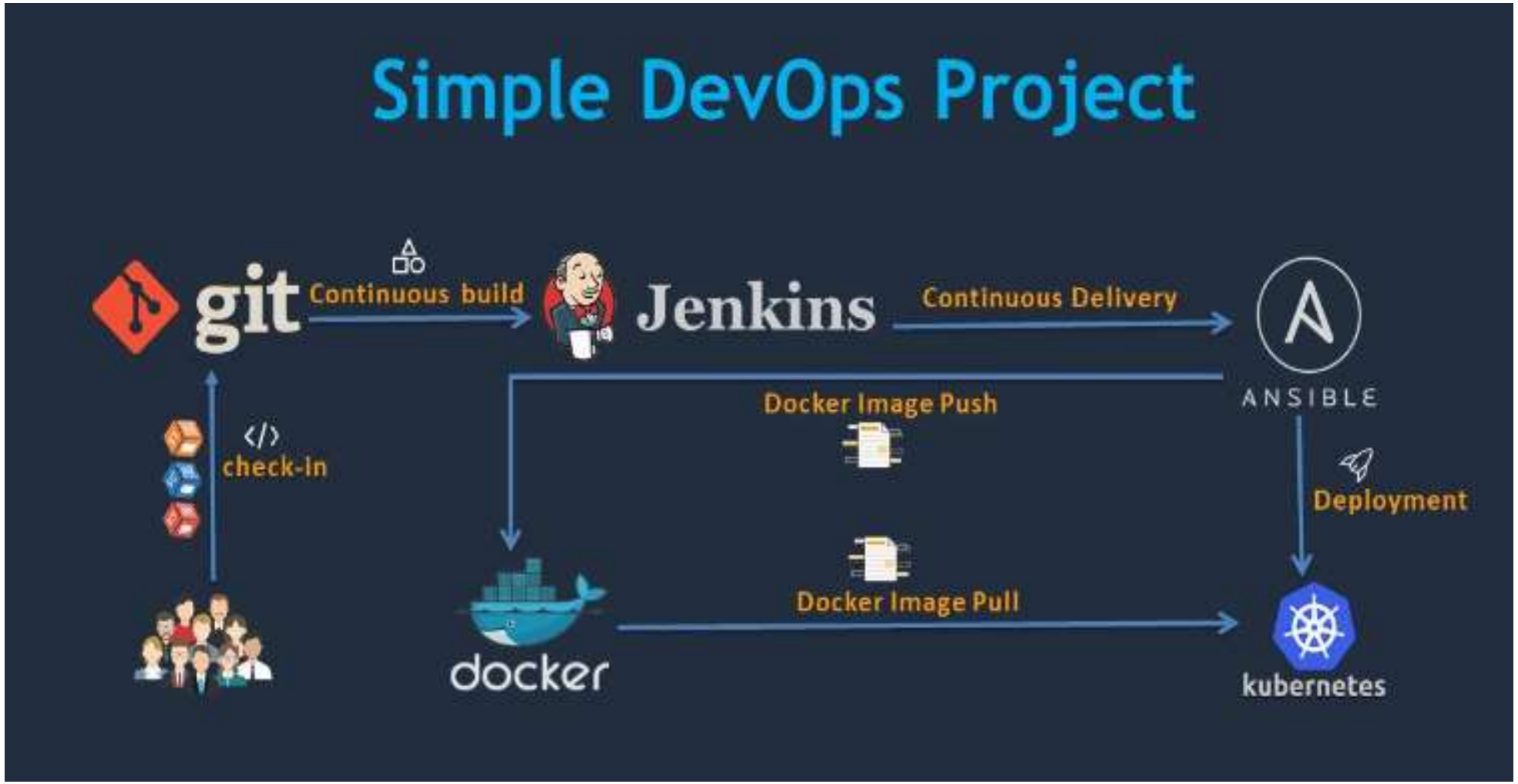
Jenkins: Jenkins je alatka koja se koristi za izgradnju CI/CD cevovoda, gde programeri mogu da prave, testiraju i primenjuju softver.

Kubernetes: Organizator kontejnera, Kubernetes se često koristi u DevOps-u.

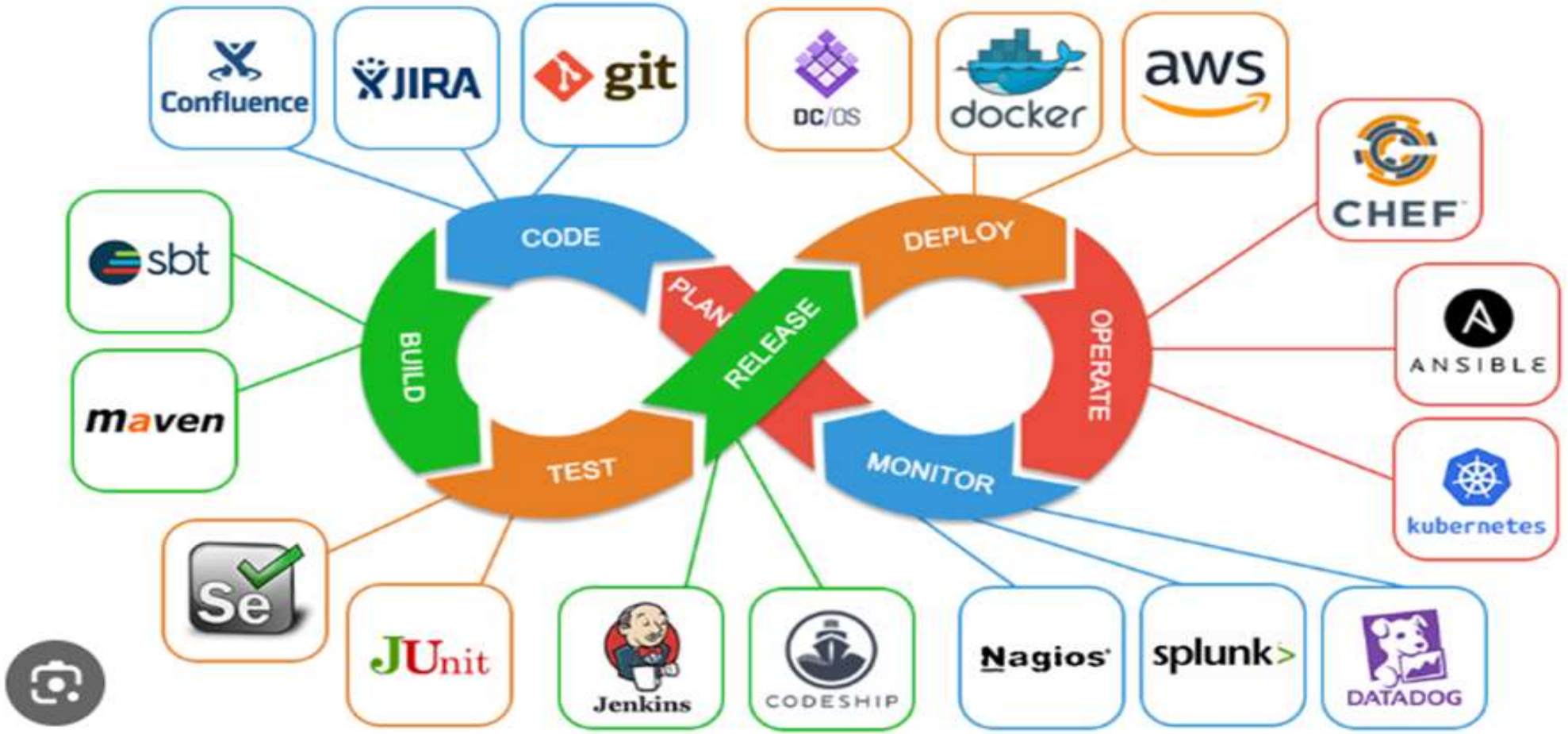
Tehnologije i alate treba da poznaješ ili naučiš?

Mi ćemo nabrojati samo neke, kao što su:

- **Alati za upravljanje izvornim kodom** – kao GitHub koji ti omogućava uvid u promene koje su napravljene u izvornom kodu.
- **Alati za implementaciju** – kao **Terraform**, sa infrastrukturom otvorenog koda koja omogućava efikasnu transformaciju, razvoj i verzioniranje.
- **Cloud alati** – uz Lambdu, alat pogodan za izvršavanje koda u različitim pozadinskim aplikacijama ili uslugama.
- **Alati za kontinuiranu integraciju** – a ovde je jedan od najkompletnijih **Jenkins**, jer ti omogućava da izvodiš testove u realnom vremenu i distribuiraš kod različitim timovima.
- **Alati za praćenje** – Grafana, jer omogućava razvoj aplikacije i uvide u potencijalna ponavljanja kvarova, kao i Prometej, alat za praćenje otvorenog koda, posebno kada je reč o kontejnerima i mikroservisima.
- **DevOps kontejneri** – omogućavaju odvajanje aplikacija od okruženja u kojima se primenjuju, uz alate kao **Kubernetes** (za implementaciju i upravljanje sistemima razvijenim za Linux kontejnere) i **Doker** (za automatizaciju i standardizaciju implementacije aplikacija).
- **DevOps klasteri** – Kops, sa komandnom strukurom za dizajniranje, ažuriranje i konfigurisanje klastera. Mreže komandne linije – Netstat, Lotop, Strace, lonotifywait, Curl, Ngrep...
- **Mrežni protokoli** – Telnet, Ping, NFS...


Simple DevOps Project
git
Continuous build
Jenkins
Continuous Delivery
ANSIBLE
Docker Image Push
check-in
Deployment
Docker Image Pull
docker
kubernetes

Potrebni alati , program da bi ste postali DevOps inženjer su



DevOps putovanje počinje obezbeđivanjem računarskih resursa potrebnih za pokretanje koda programera putem nepromenljivih implementacija.

Terraform i ***CloudFormation*** se koriste za obezbeđivanje infrastrukture, a ***Ansible*** se koristi za njeno konfigurisanje.

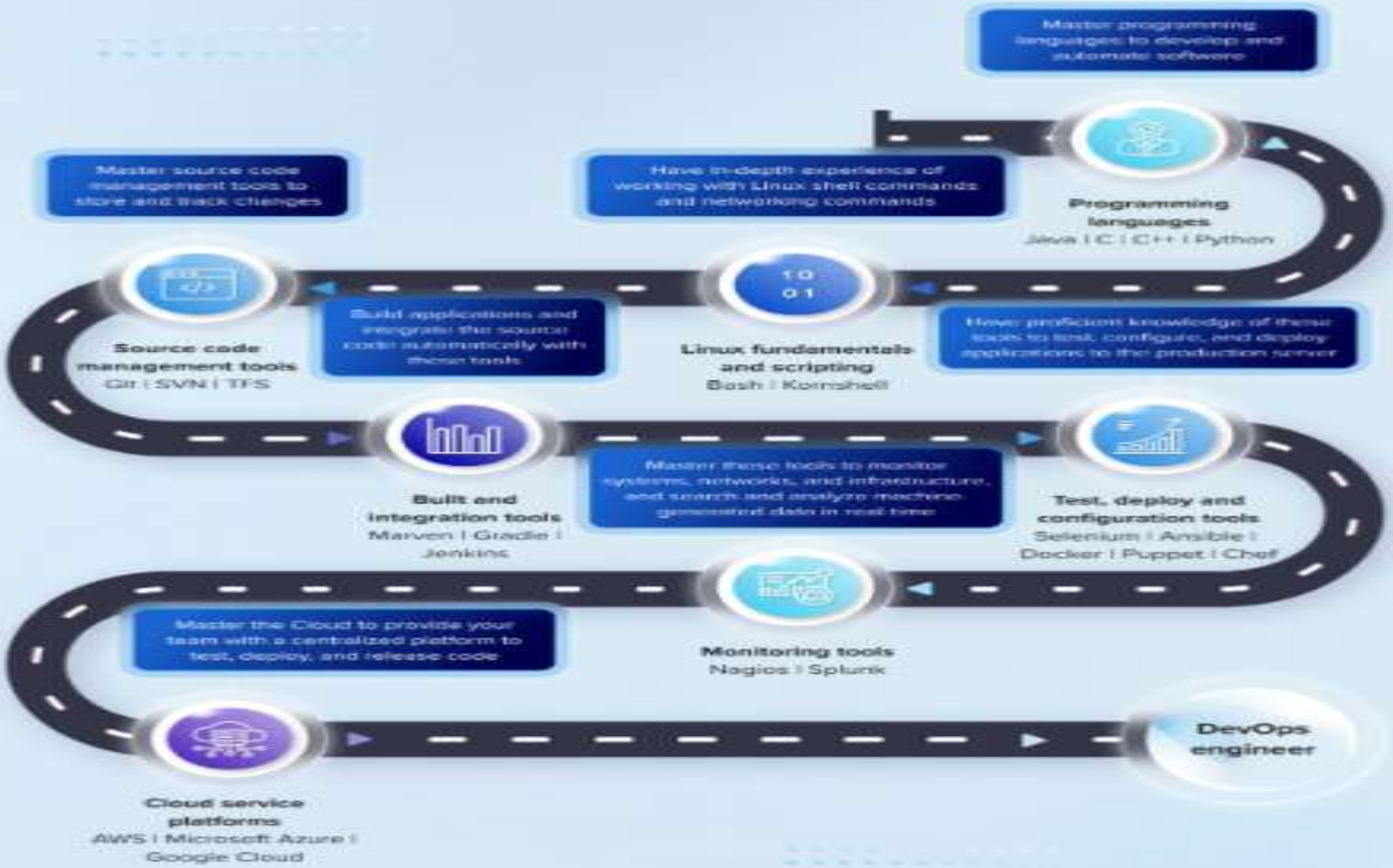
Faza verzionisanja obezbeđuje kontrolu izvornog koda, održavajući zajedničku bazu koda na centralizovanoj lokaciji. Uz ***Git***, svi proizvodni artefakti su verzionisani, podložni praćenju, pregledu i istoriji promena.

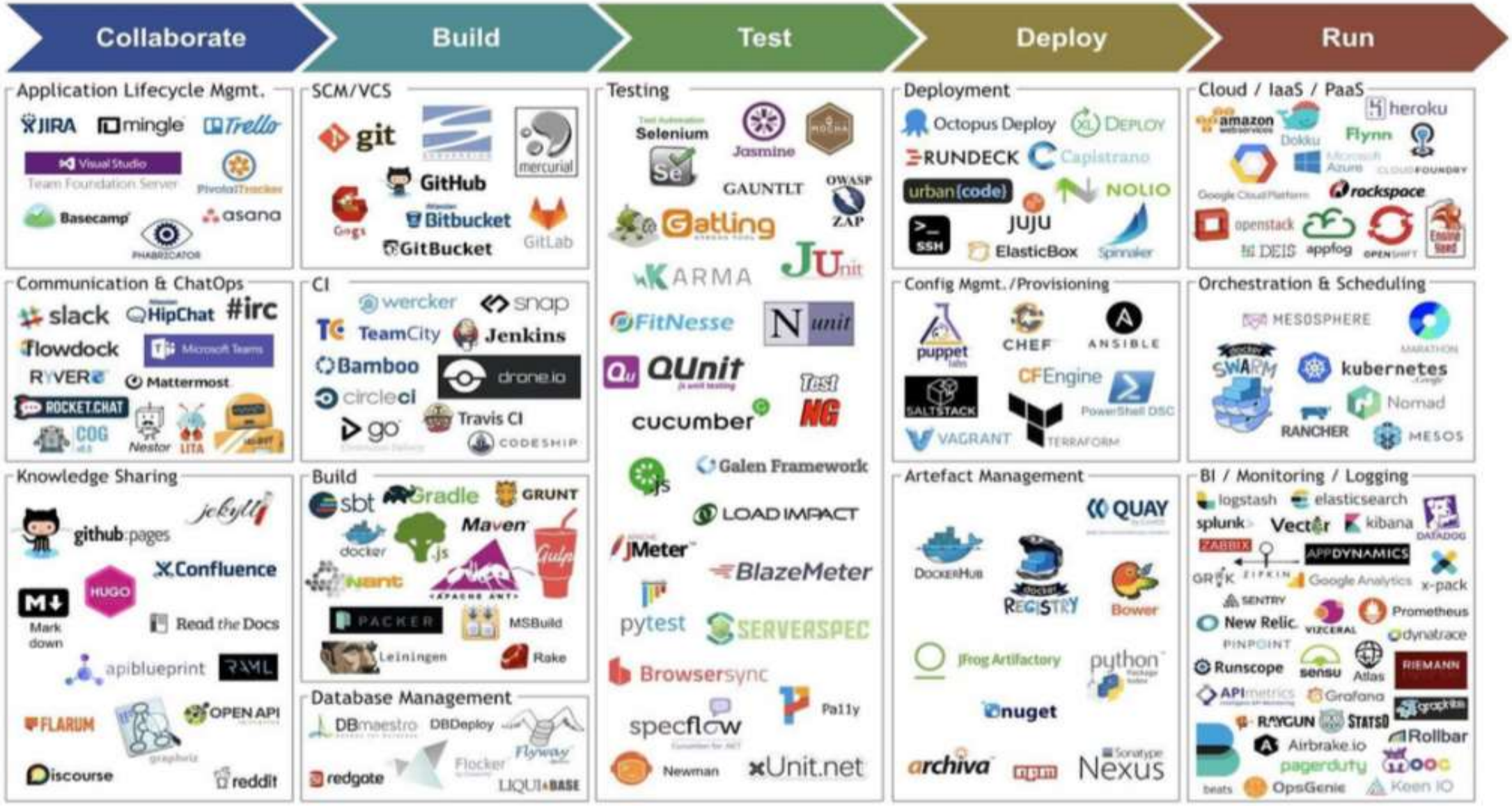
Morate da vežbate Git da biste bili sigurni u forkiranje spremišta, kreiranje grana, spajanje promena, odobravanje zahteva za povlačenjem itd.

U fazi implementacije, morate savladati pristup inženjeringa mikroservisa. Pošto je individualna implementacija softverskih komponenti velika muka, DevOps inženjeri pakuju, postavljaju i dosledno pokreću različite aplikacije. Stoga smo pakovanje za učenje postavili kao cilj koji morate imati.


Junior DevOps Roadmap
1
2
3
4
5
6
Provision
Version
Package
Deploy
Run
Observe
Terraform
GitHub
GitLab
Lambda
docker
GitHub
GitLab
ECS
kubernetes
Grafana

Middle and Senior DevOps level





https://roadmap.sh/devops