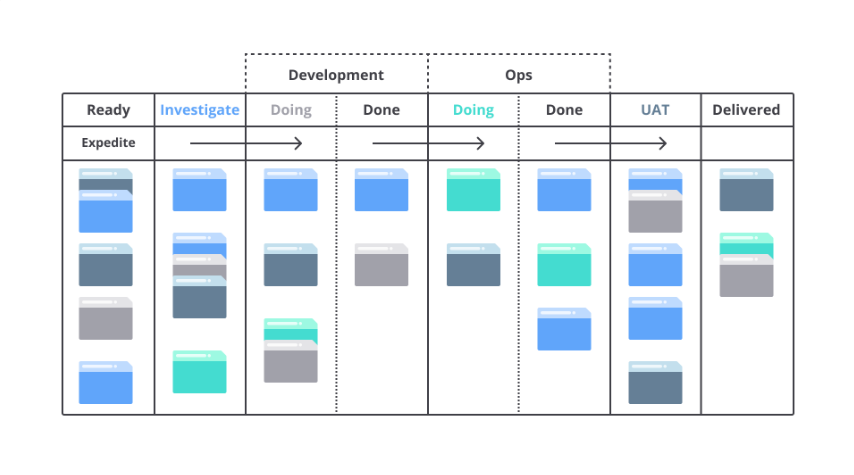
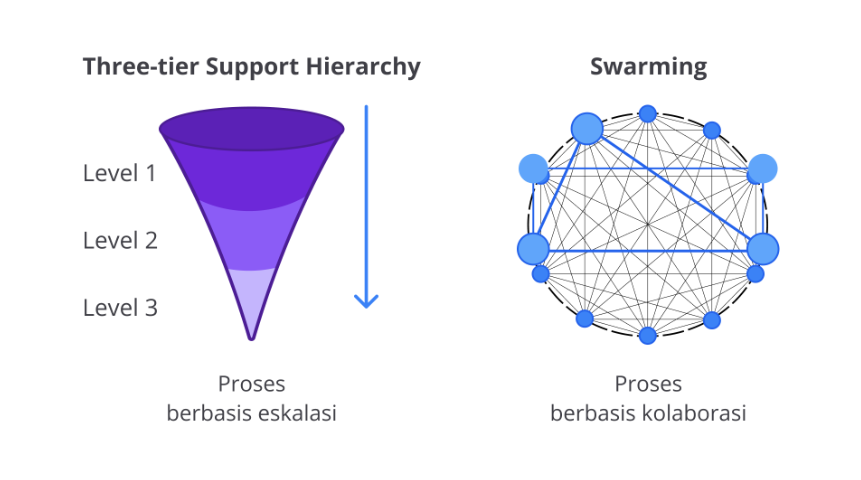
Belajar Devops Dasar – Dicoding (Trial 15 hari)

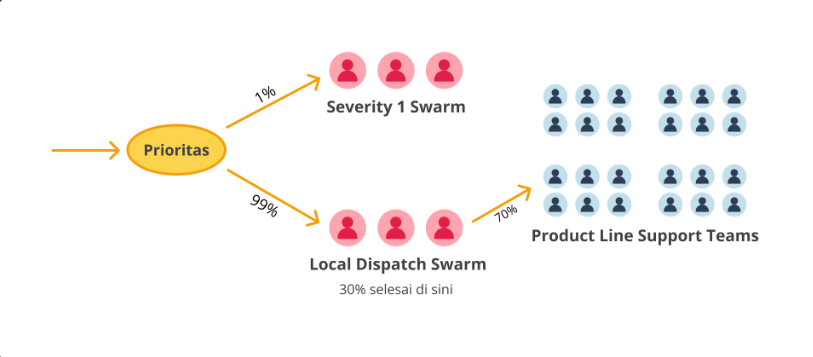
**Day 1**



1. Ready = proses dimana aplikasi siap untuk dikerjakan oleh developer
2. Investigate = proses yang memuat spesifikasi apa saja yang harus ada di aplikasi seperti fitur apa saja yang perlu di deploy, kebutuhan platformnya seperti server dll
3. Development = proses dimana code untuk aplikasi mulai ditulis oleh developer
4. Ops = Proses dimana arsitektur sudah direncanakan dan code yang ditulis developer siap untuk di deploy
5. UAT = Setelah code sudah dideploy ke platform user, dilakukan testing internal sebelum aplikasi di deliver ke customer.
6. Delivered = Proses dimana setelah semua proses dan UAT testing telah selesai dilakukan dan lulus dari proses pengujian



Dalam dunia IT, DevOps merupakan sebuah kombinasi dari developer dan IT Operations, cukup banyak sebuah perusahaan menggunakan Three-tier Support Hierarchy untuk menangani sebuah insiden yang terjadi, tapi pada sebenarnya proses tersebut menjadi penghambat bagi IT Operations karena pada proses tersebut memungkinkan terjadinya ‘Bouncing Incident’ yang dimana Ketika insiden lanjut di eskalasi ke level 2, namun level 2 masih memerlukan informasi dari level 1, dan begitu seterusnya, dengan adanya prinsip tersebut, muncullah prinsip ***Swarming,*** yang dimana penanganan insiden langsung ke ahlinya tanpa melewati eskalasi dari level ke level, untuk mengurangi antrean dan handoff yang menghambat proses penanganan insiden



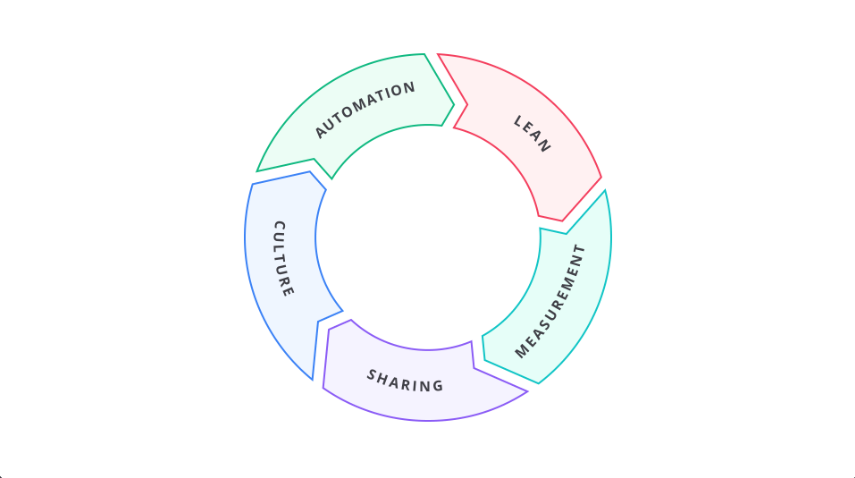
Pada metode swarm ini terdapat mekanisme kolaborasi antar tim, yang dibagi menjadi beberapa kategori

1. Severity 1 Swarm = Tim ini berisi orang orang yang sangat paham/ spesialis terhadap suatu insiden atau topik yang ditangani dengan perwakilan setiap tim , tim ini akan menangani insiden yang benar benar kritis/krusial, sehingga dari semua insiden yang masuk mungkin tim ini hanya menangani 1-5% saja karena hanya yang kritis/krusial saja, sisanya akan dilempar ke Local Dispatch Swarm
2. Local Dispatch Swarm = Tim ini akan menerima sebagian besar insiden yang terjadi, dan mereka akan memfilter insiden apa saja yang dapat mereka selesaikan dengan cepat/yang paling cepat dapat solusi sisanya akan dilanjutkan ke Product Line Support Teams/Cuscare
3. Backlog Swarm = Tim ini hampir mirip dengan Severity 1 Swarm, karena berisi orang orang terampil pada bidangnya, mereka akan menangani insiden yang tidak bisa ditanganin oleh Local Dispatch Swarm dan Product Line Support Teams

Dengan proses Swarming ini kita dapat benefitnya seperti

1. Tidak adanya insiden yang on progress berkepanjangan
2. Antar tim dapat berbagi pengetahuan saat troubleshoot insiden
3. Memungkinkan berbagi pengetahuan antar tim

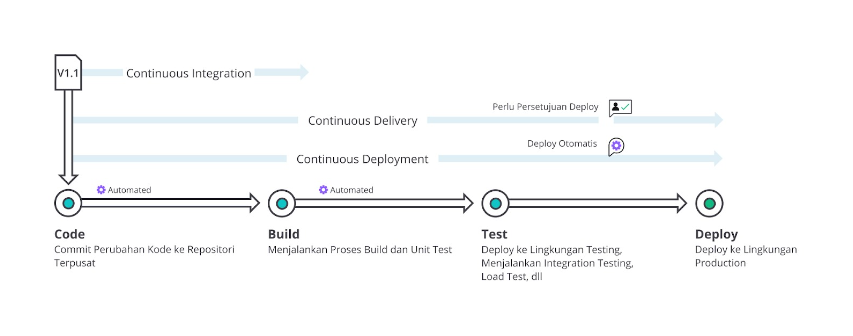
**Day 2**



1. Culture = Dalam mengembangkan DevOps, kita perlu membangun sebuah kebiasaan yang didalamnya mengikutsertakan tim Developer dan IT Operation sehingga kedua tim dapat berkolaborasi dengan baik dalam mengembangkan sebuah aplikasi sehingga aplikasi dapat terdeliver dengan baik ke user.
2. Automation = Proses otomasi dalam mengembangkan sebuah aplikasi, misalnya seperti code integration, code review, monitoring alert, dll
3. Lean = Pada proses ini setiap proses akan diusahakan efisien, meminimalkan code wip atau pekerjaan coding yang berlebih, mengurangi penyerah terima kerjaan ke tim lain, mengurangi waktu tunggu untuk pindah ke fase development selanjutnya contoh dari proses code masuk ke repositori, bagaimana caranya mempercepat untuk di deploy dan dijalankan
4. Measurement = Proses pengukuran proses development yang sedang terjadi, misal berapa lama proses developmentnya, berapa banyak bug yang muncul, berapa lama bisa merecover saat terjadi kegagalan pada sistem
5. Sharing = Proses untuk berkolaborasi antar tim dengan berbagi pengetahuan dan berbagi tanggung jawab sehingga proses kolaborasi terjadi dengan baik.



1. Code = Developer menuliskan dan mengembangkan kode aplikasi dalam bahasa tertentu dan tentu saja pada environment development agar tidak mengganggu aplikasi yang sedang berjalan
2. Build = Setelah kode sudah dikembangkan dan di simpan kedalam repository, langkah selanjutnya adalah proses build/compile untuk mengubah kode menjadi sebuah aplikasi yang matang/jadi
3. Test = Tahap pengujian setelah aplikasi di build, pada tahap ini seharusnya menjadi tahap yang krusial karena kita dapat menguji apakah aplikasi terdapat bug/error atau tidak, sehingga saat nanti kita ingin delivery, produk sudah stable
4. Release = Pada tahap ini aplikasi sudah bisa dibilang stable, karena bug/error sudah dieliminasi saat pengujian dan juga pemberian nomor versi perilisan biasanya
5. Deploy = Proses migrasi dari environment development ke production
6. Monitor = Proses setelah aplikasi didelivery, kita melakukan monitor by system, seperti alert atau incident, seperti menggunakan Grafana



**Continuous Integration**

**Continuous integration (CI)** merupakan praktik pada proses pengembangan aplikasi di mana Developer dengan rutin dan teratur memasukkan (commit) atau menggabungkan (merge) setiap perubahan kode (code changes) mereka ke sebuah repositori terpusat (central repository) dan/atau ke mainline trunk (seperti branch master/main), setelah itu proses build dan unit test secara otomatis pun dijalankan.

**Continuous Delivery**

**Continuous delivery (CD)** adalah praktik pada proses pengembangan aplikasi di mana perubahan kode (code changes) secara otomatis dipersiapkan sebelum nantinya dikirim ke lingkungan production.

Continuous delivery merupakan teknik lanjutan dari continuous integration. Jika di CI hanya sampai proses build dan unit test, di CD ini prosesnya hingga deploy semua perubahan kode ke lingkungan testing, staging (pre-production), dan/atau production. Namun, untuk bisa men-deploy ke production, perlu melalui persetujuan manual (manual approval) terlebih dahulu, entah itu oleh Developer yang lebih senior, manajer, atau siapa pun yang berhak.

**Continuous Deployment**

Continuous delivery dan continuous deployment pada hakikatnya adalah proses yang “serupa tapi tak sama”. Perbedaannya, continuous delivery memiliki proses persetujuan manual (manual approval) sebelum aplikasi di-deploy ke production, sementara continuous deployment tidak memiliki hal tersebut.

Jadi, dengan continuous deployment, proses deploy aplikasi ke lingkungan production berlangsung secara otomatis tanpa ada persetujuan eksplisit dan intervensi manusia. Dengan begitu, continuous deployment memungkinkan pemberian feedback yang berkelanjutan oleh pengguna setiap kali suatu fitur atau update sampai ke perangkat mereka.

Tools untuk DevOps

Code = Github, Gitlab, BitBucket untuk version controller

Build = Jenkins, Travis CI, CircleCI

Test = Gremlin, Testable, LoadRunner, UFT One

Release = Docker Hub, JFrog Artifactory

Deploy = Heroku, Netlify, Github Pages

Monitor = Prometheus, Grafana, Elastic Stack, Dynatrace

Day 3

**Infrastructure as Code**

Dalam mengembangkan sebuah aplikasi kita perlu melakukan hal efisien, yang biasa digunakana pada tahap devops adalah Infrastructure as Code yang dimana proses scripting yang membantu devops engineer untuk membuat environment secara cepat dan handal

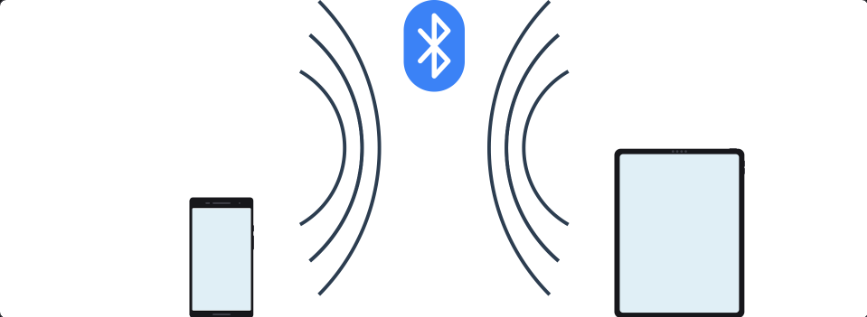
1. **Infrastructure Automation** = proses otomasi dalam penerapan infra, bisa menggunakan tools seperti Terraform
2. **Configuration Management** = proses otomasi dalam pengkonfigurasian sebuah sistem, bisa sistem operasi ataupun server sendiri, scripting seperti ini biasanya digunakan pada proses yang repetitive, tools yang digunakan biasanya Chef, Puppet, Ansible
3. **Architecture Microservice** = proses membangun aplikasi dengan pendekatan desain menggunakan sekumpulan service yang dipisah/dibagi menjadi kecil, terpisah tetapi masih menjadi satu kesatuan sehingga pada saat development tidak mengganggu satu sama lain, tools yang biasanya digunakan adalah Kubernetes, Openshift, Nomad
4. **Serverless** = pendekatan komputasi yang dimana saat kita mengembangkan sebuah aplikasi tanpa perlu memikirkan infrastruktur dan server

**Belajar Jaringan Komputer Pemula – Dicoding Trial 15 hari**

Day 3

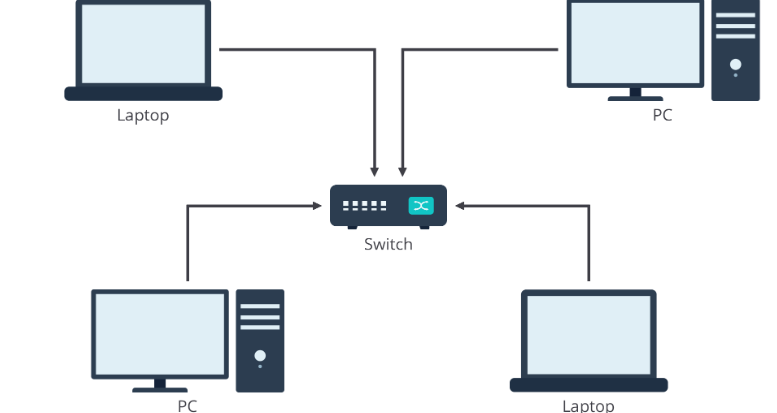
**Tipe-tipe Jaringan**

**Personal Area Network (PAN)**



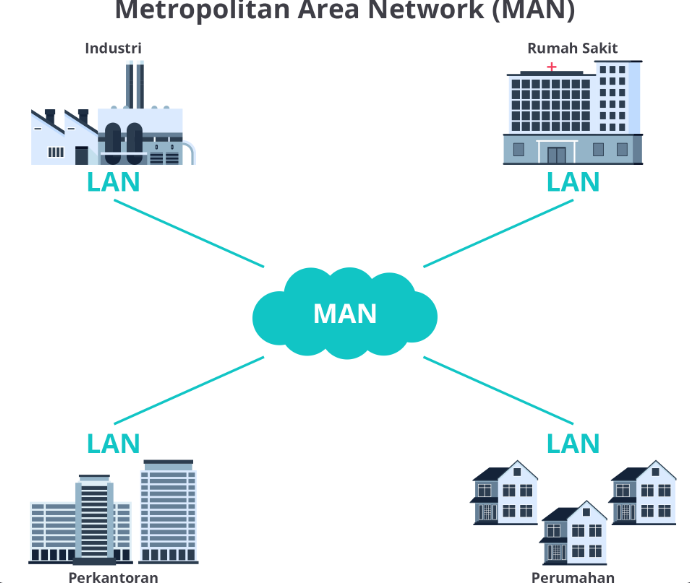
Tipe jaringan dengan cakupan terkecil dan sangat pribadi, biasanya hanya antar personal, seperti Bluetooth

Local Area Network (LAN)



Tipe jaringan yang mencakup area relative kecil seperti pada satu kantor, satu gedung untuk menghubungkan device secara local

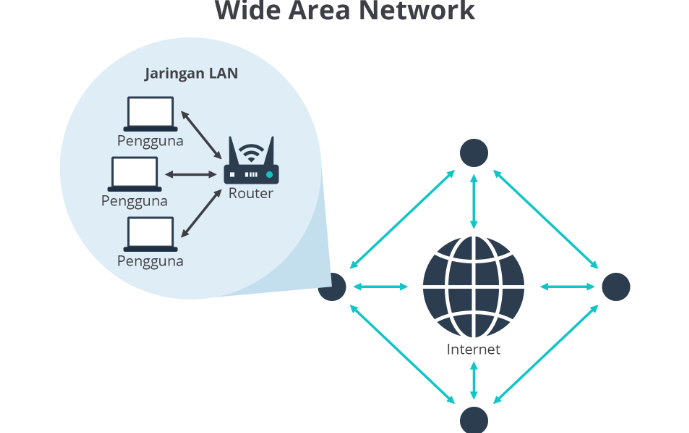
**Metropolitan Area Network (MAN)**



Tipe jaringan ini biasanya mencakup antar gedung, antar wilayah, seperti

* Institusi pemerintahan yang membutuhkan koneksi tepercaya untuk support kebutuhan data dan sistem.
* Perusahaan yang memerlukan koneksi berkapasitas besar dari atau ke DC (Data Center) dan DRC (Disaster Recovery Center).

**Wide Area Network (WAN)**

****

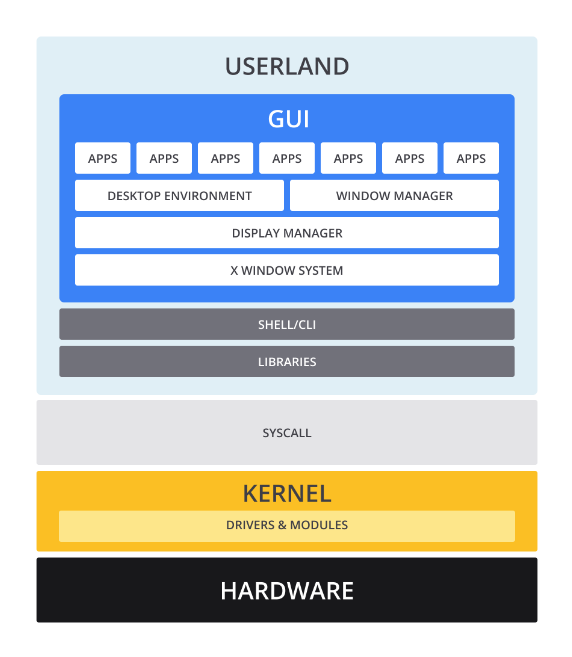
Tipe jaringan ini memiliki cakupan yang sangat luas dan cukup rumit, biasanya antar provider

Day 4

**Menjadi Linux System Administrator**

**Arsitektur Linux**

Secara garis besar, arsitektur dari Linux dapat digambarkan menjadi seperti dibawah ini.

****

1. Userland = Ini merupakan bagian yang bisa digunakan oleh user, misalnya seperti GUI, Shell, Libraries, seperti pada windows desktop pada umumnya
   1. GUI = Bagian ini merupakan interface graphical yang ada pada linux, yang terdiri dari X Windows System, Display Manager, Desktop Environment, Window Manager, dan beberapa aplikasi pengguna
   2. Shell = Merupakan sebuah Command Line yang bertugas untuk menerima semua inputan perintah user seperti pada CMD di windows, seperti contoh commandnya:
      1. ls, rm, mv, ln = pengelolaan berkas, ls melihat file, rm menghapus file, mv mengubah atau memindahkan/cut file, ln menghubungkan relasi file pada direktori berbeda
      2. wget = mengunduh berkas dari library di cloud
      3. curl = mengirim atau mengambil data melalui URL
      4. ssh = perintah untuk remote ke server
      5. ping = memeriksa jaringan
   3. Libraries = Ini menjadi komponen pendukung pada linux untuk menjalankan aplikasi pada linux, tanpa library terkait aplikasi tidak dapat dijalankan
2. Kernel = Komponen ini berfungsi penting pada sistem operasi linux, karena menjadi penengah antar user dengan sistem, bagian ini memiliki 2 komponen utama yaitu Syscall & Kernel itu sendiri
   1. Syscall = merupakan komponen yang memiliki peran untuk menerima intruksi dari pengguna untuk dijalankan, jadi seperti syscall adalah backendnya, dan shell/CLI merupakan frontendnya, seperti contoh saat kita menginputkan perintah *ls* maka syscall akan memanggil library API Glibc
   2. Kernel = merupakan inti dari sebuah sistem operasi, fungsi utama kernel adalah untuk mengelola sumber daya (RAM, CPU, Storage, dll), kernel juga bisa diibaratkan seperti asisten pribadi kita, yang memiliki tugas
      1. Menyampaikan pesan, dan permintaan dari user
      2. Mengelola dan mengingat lokasi penyimpanan barang (dalam case ini storage management)
      3. Mengatur jadwal user, seperti siapa saja yang dapat memiliki akses, berapa lama waktu, dan kapan waktunya
3. Hardware = Hardware ini merupakan perangkat keras yang dapat menjadi platform bagi sistem operasi linux, pada linux kita dapat mengetahu beberapa informasi hardware, seperti
   1. lspci = Menampilkan daftar hardware yang menggunakan BUS PCI
   2. lsusb = Menampilkan daftar hardware yang terpasang menggunakan USB
   3. cat /proc/cpuinfo atau lscpu = Menampilkan informasi CPU
   4. cat /proc/meminfo = Menampilkan informasi computer
   5. fdisk -l = Menampilkan partisi hardisk
   6. dmidecode = Menampilkan informasi lengkap terkait hardware komputer