Os 12 fatores + CI/CD

7COMm
Serviços e Soluções em TI

Heroku

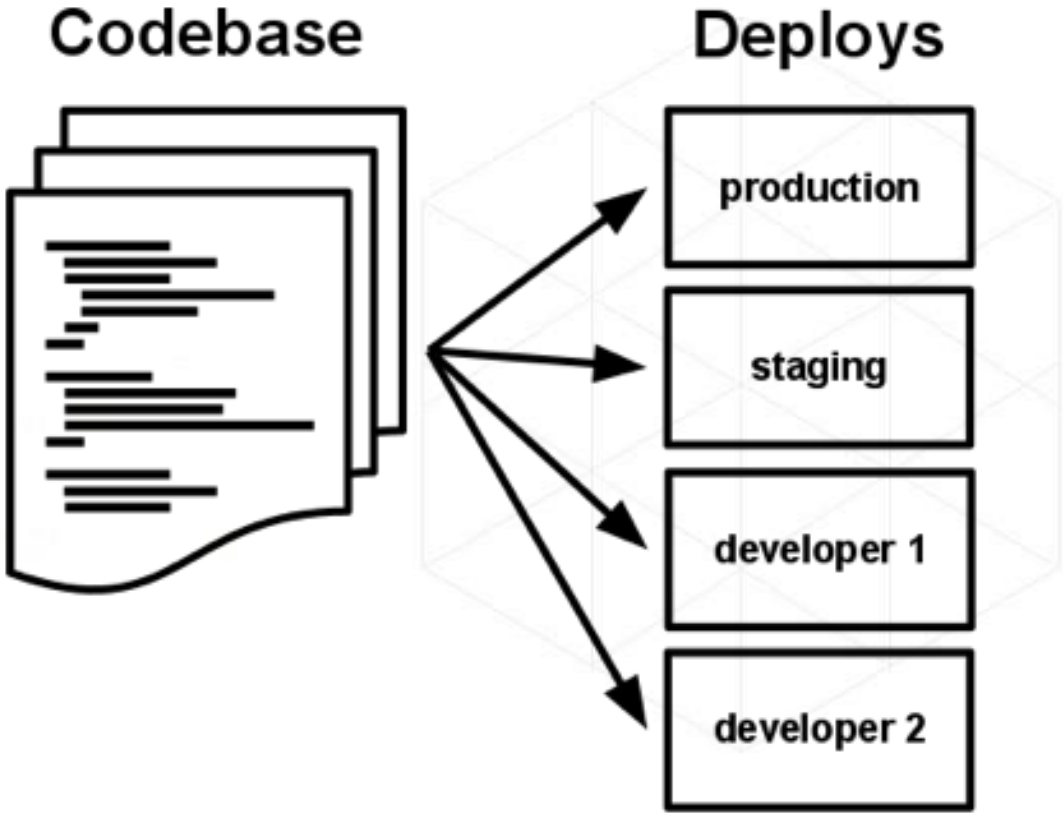
- Primeira empresa a entrar de cabeça na "nuvem pública"
- A promessa era não se preocupar mais com infraestrutura
- Desenvolvedores não tinham, então, o conhecimento para este tipo de desenvolvimento, focado na nuvem
- O time de engenheiros criou um manifesto com 12 fatores, com a finalidade de mostrar como criar aplicações *cloud-first*

Os 12 fatores

- Utilizam formatos declarativos para automatizar a configuração inicial e minimizar custos
- Adequados para implantação em plataformas de nuvem
- Minimizam a divergência entre ambientes de desenvolvimento e produção
- Facilidade na escalabilidade sem significativas mudanças em ferramentas ou práticas de desenvolvimento

1- Base de código

- Todo seu código deve ser gerenciado por um sistema de controle de versão
- Não deve haver mais de uma base de códigos. Desta base são gerados vários *deploys*
- Um *deploy* é uma versão do código sendo executada em ambientes distintos



2- Dependências

- Declare e isole as dependências
- Nunca deve-se confiar na existência de pacotes *implícitos*
- Todas as dependências são declaradas através de um manifesto
- Simplifica a configuração da aplicação para novos desenvolvedores

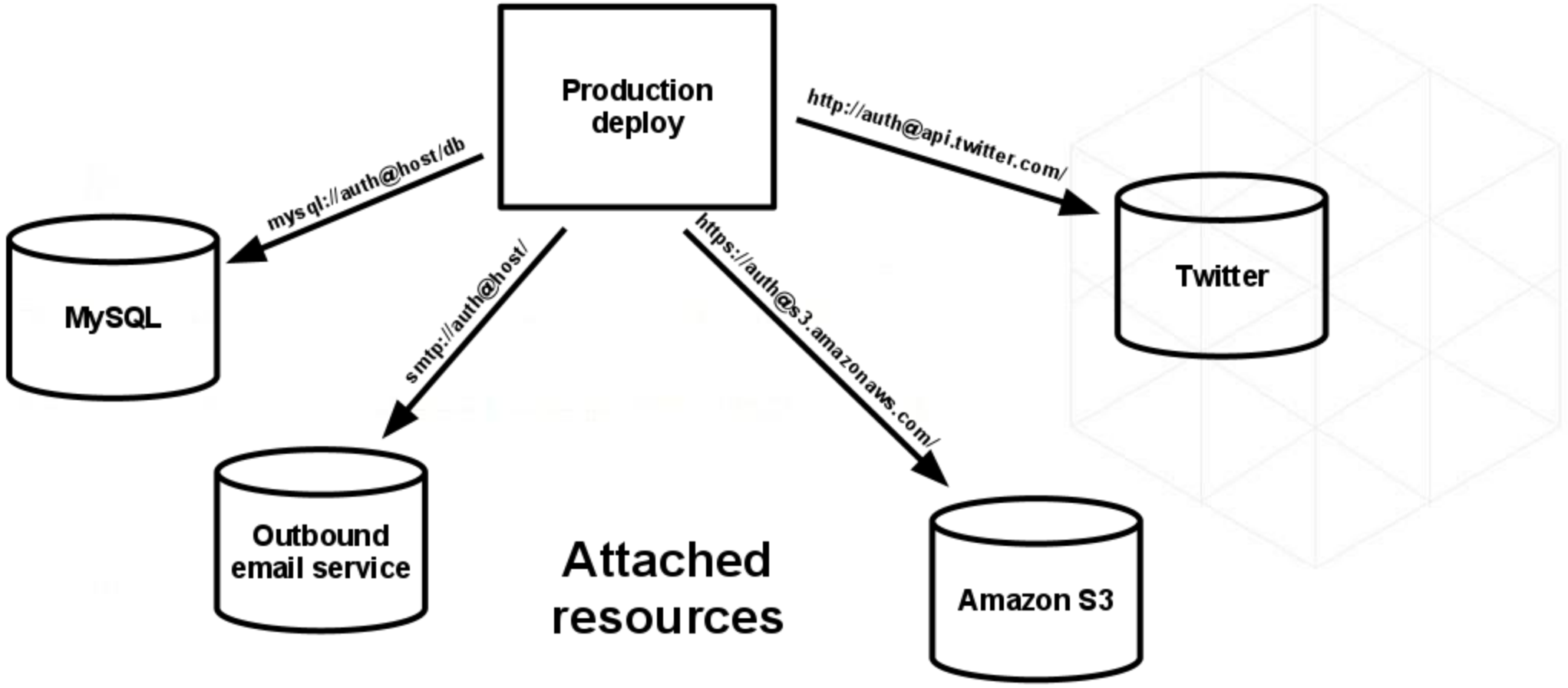
3- Configurações

- Tudo que pode variar entre *deploys*
  - Bancos de dados; credenciais de acessos; hosts, etc.
- Não utiliza-se configurações em constantes
- Pode-se utilizar arquivos de configuração não versionados
- Utiliza-se variáveis de ambiente gerenciadas independentes em cada deploy

4- Serviços de apoio

- Qualquer serviço que a aplicação consuma via rede, como parte de operação normal
  - Bancos de dados; mensageria; smtp; etc...
- Serviços localmente gerenciados, assim como serviços de terceiros devem ser tratados indistintamente
- Um servidor local deve poder ser trocado por um servidor terceiro sem alteração no código

4- Serviços de apoio



5- Build, release, run

- Um deploy deve ser gerado em três estágios
  - Build: converte um repositório em executável
  - Release: combina o build com a configuração do deploy
  - Run: roda a aplicação no ambiente de execução
- Deve-se separar estritamente os três estágios
  - Impossível alterar o código em tempo de execução
- Cada release deve ter um único ID como um timestamp ou uma tag, em caso de docker

6- Processos

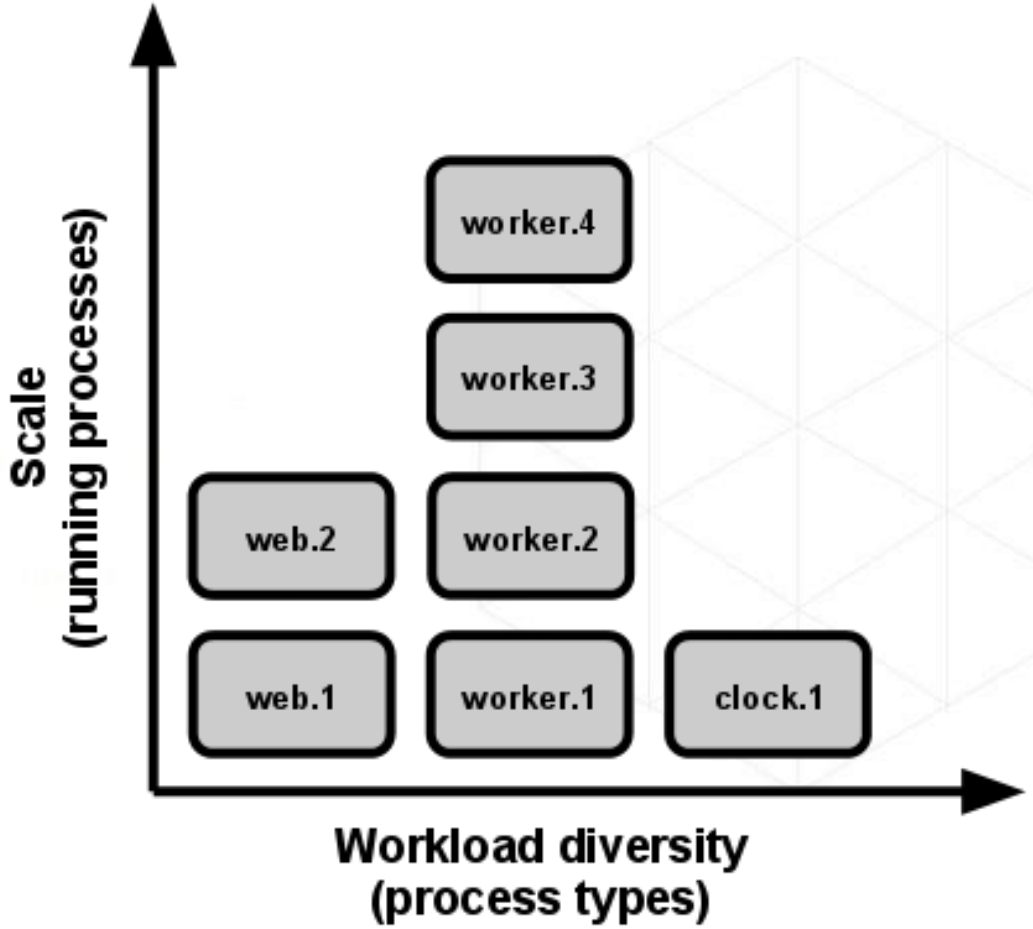
- A aplicação deve ser executada como um ou mais processos que não armazenam estado
- Quaisquer dados que devem ser persistidos devem ser armazenados em um serviço de apoios statefull
- Sessões persistentes em aplicações web nunca devem ser utilizadas. Dados de sessão podem ser considerados para um datastore, como um memcached ou Redis

7- Vínculos de porta

- Aplicação auto contida
- Não depende de servidores web externos
- São comunicáveis a partir de portas de rede
- Qualquer aplicação pode se tornar um serviço de apoio pra outras

8- Concorrência

- Cada aplicação corresponde a um ou mais processos em memória
- Cada funcionalidade da aplicação deve ser modelada de forma que seja atendida por um ou mais processos que possam escalar, reiniciar ou se clonar, quando necessário



9- Descartabilidade

- Um processo deve ser descartável
- Pode ser iniciado ou terminado sem grandes impactos no tempo de execução ou disponibilidade das funcionalidades
- Isso permite escalabilidade e reconfiguração sem grandes impactos

10- Desenvolvimento/Produção semelhantes

- Manter os ambientes de desenvolvimento, testes com mesmo SO, serviços de apoio e dependências
- Gaps
  - Temporais: código leva tempo para ir pra produção
  - Papéis: dev codifica e profissional de operações implanta
  - Gap de ferramentas: recursos usados em desenvolvimento divergem dos recursos de produção, como versão de banco de dados

10- Desenvolvimento/Produção semelhantes

- Reduzir tempo entre deploys para horas ao invés de dias, semanas ou meses
- Permite que o mesmo profissional seja o responsável pela implantação da aplicação até a produção
- Nivelar os ambientes da aplicação, objetivando manter somente as diferenças de performance/volume

11- Logs

- Tratar os logs como uma visão comportamental da aplicação
- Considerados como fluxos de informaçõs ordenadas temporalmente que não devem ser um item de preocupação da aplicação
- Aplicação NÃO deve ser responsável pelo armazenamento e gerenciamento de logs

12- Processos de Admin

- Tarefas de administração devem ser executados em ambiente semelhante ao ambiente da aplicação
- Devem ser executados sobre o mesmo código-fonte, configurações e mecanismos de persistência
- Devem ser empacotadas e entregues juntamente com o código-fonte responsável pelas funcionalidades
- https://12factor.net/pt_br/

CI/CD

7COMm
Serviços e Soluções em TI

What is Continuous Integration

Coninuous Integration

- Permite que tanto o processo de desenvolvimento quanto entrega seja mais ágil
- Automação para que sempre que houver uma mudança, ela seja integrada, testada e implementada
- Todas essas mudanças são realizadas no mesmo local, permitindo a integração de maneira mais ágil

Coninuous Delivery

- Integração contínua e implantação em conteiners em seguida implantado em produção
- Mesmo que necessite de ação humana, ele se torna automatizado quando colocado no ar de maneira integrada e completa

Coninuous Deployment

- Processo a mais no continuous Delivery que permite que as entrega em ambiente produtivo de forma 100% automatizada

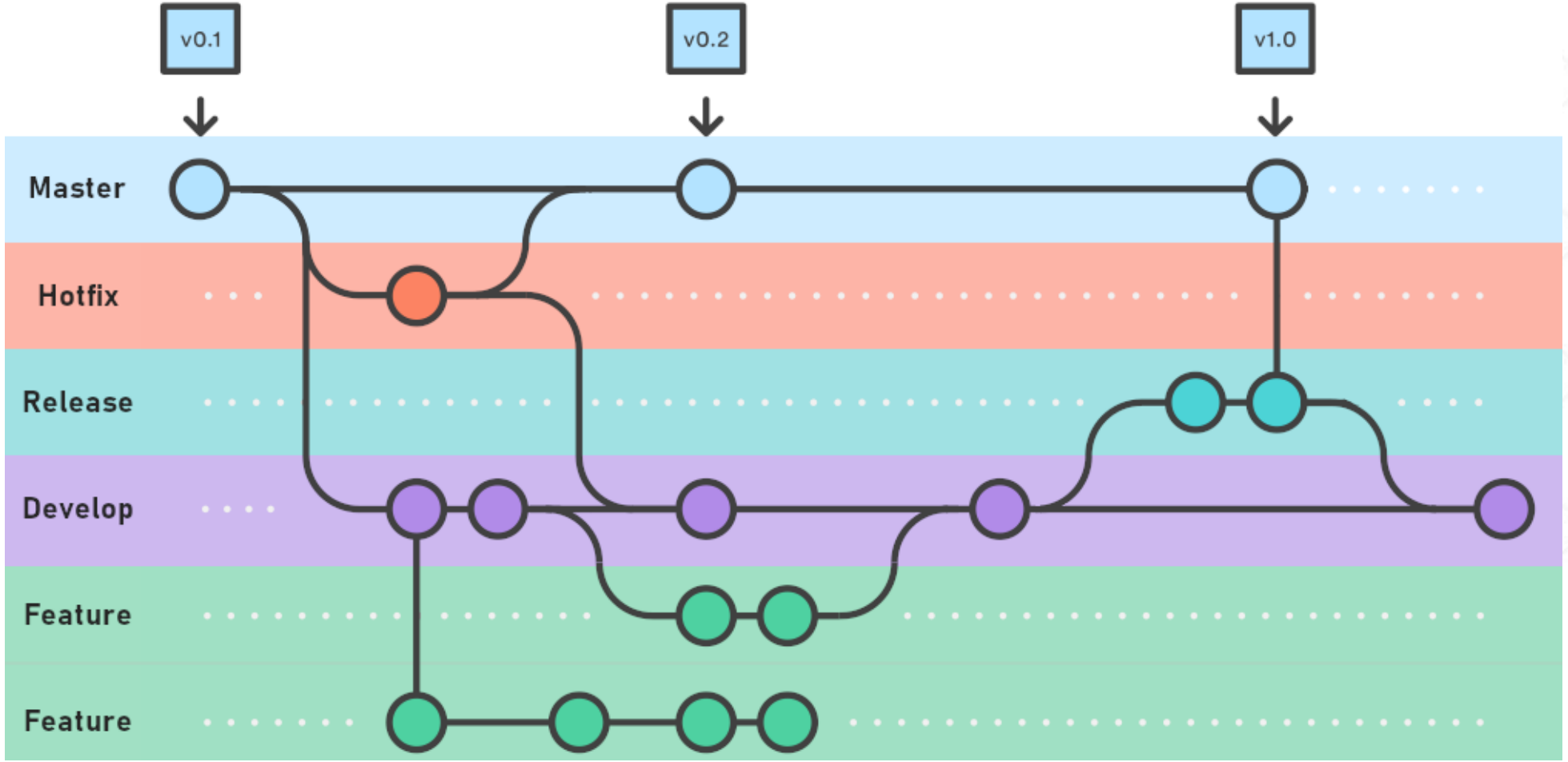
CI/CI - Melhores estratégias

- Alinhamento do processo de desenvolvimento com os times
- Mapeamento das branches do git para os ambientes
- Disparo de Job de CI para cada Pull Request e CD para cada alteração nas branches

Git Flow

- Fluxo de trabalho com o Git que facilita o desenvolvimento
- Branches estabelecidas como padrão (master, develop, feature, hotfix e release)

Git Flow



Git Flow

- Master
  - Principal branch
  - Versão de publicação do código em produção
  - Somente se interage com ela a partir de um hotfix ou release
- Develop
  - Serve como uma versão com todos os últimos desenvolvimentos
  - Cópia da branch principal, com funcionalidades ainda não entregues
  - Base do desenvolvimento de novas features

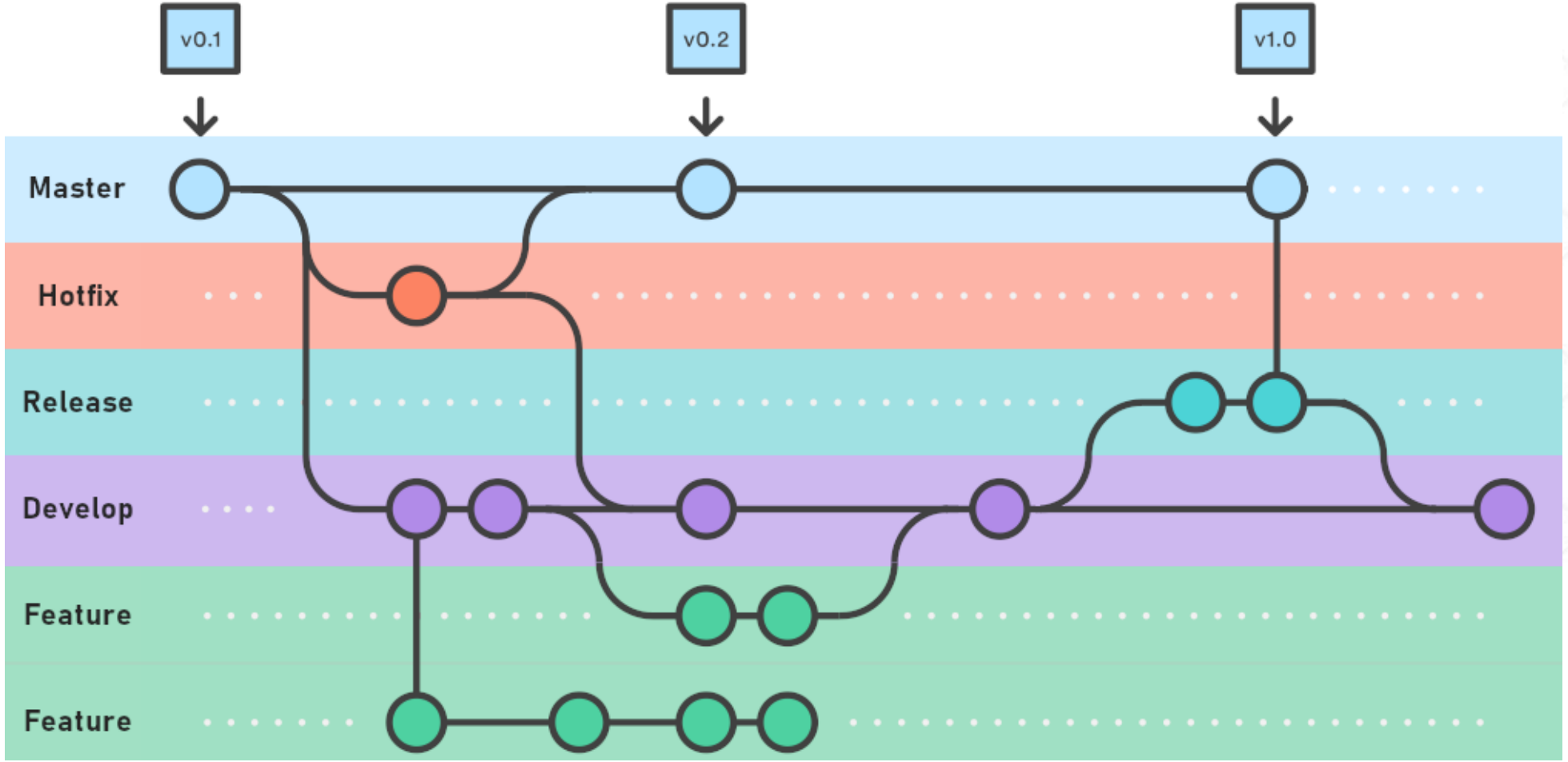
Git Flow

- Feature
  - Branch temporária
  - Contém uma funcionalidade específica
  - Convenciona-se utilizar o nome **feature/nome_do_recurso**
- Hotfix
  - Branch temporária
  - Utilizada para correções em ambiente de produção
  - Após aplicada a correção, deve-se fazer o merge dela pra branch develop

Git Flow

- Release
  - Branch de lançamento
  - Reune-se o que está na branch develop e faz o merge para a master
  - A partir dela é criada uma versão tagueada do projeto

Git Flow



Jenkins

- https://www.jenkins.io/
- Servidor para automação de código aberto
- Projeto iniciado em 2004
- Orquestra todo o pipeline de entrega de software

Jenkins – Pipeline de Automação

- Conjunto de plugins que suportam a implementação de CI/CD
- Pode ser tanto declarativo quanto baseado em scripts
- https://www.jenkins.io/doc/book/pipeline/

Jenkins – Pipeline de Automação

Jenkinsfile (Declarative Pipeline)

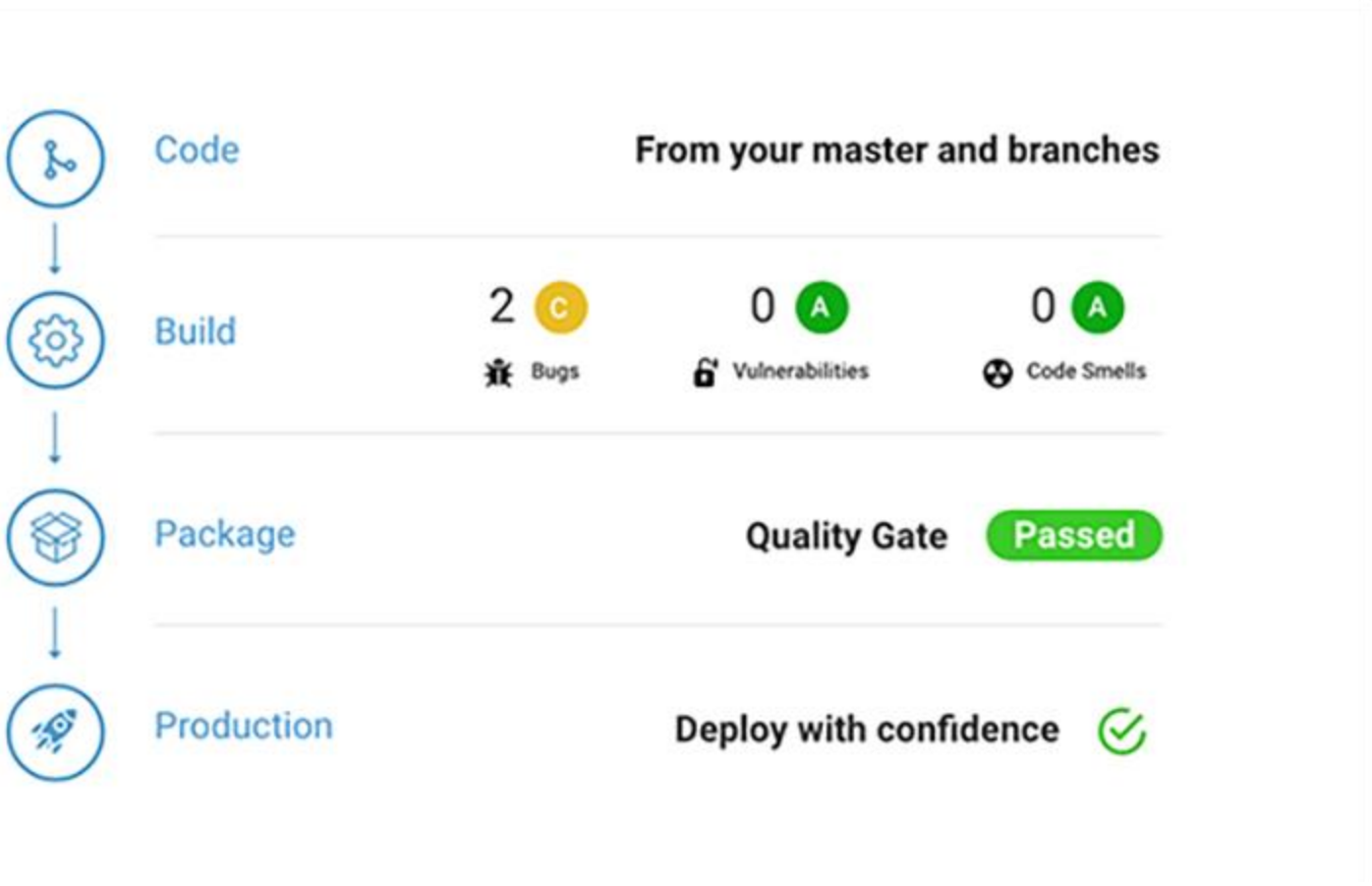
```
pipeline {
    agent any (1)
    stages {
        stage('Build') { (2)
            steps {
                // (3)
            }
        }
        stage('Test') { (4)
            steps {
                // (5)
            }
        }
        stage('Deploy') { (6)
            steps {
                // (7)
            }
        }
    }
}
```

1. Execute this Pipeline or any of its stages, on any available agent.
2. Defines the "Build" stage.
3. Perform some steps related to the "Build" stage.
4. Defines the "Test" stage.
5. Perform some steps related to the "Test" stage.
6. Defines the "Deploy" stage.
7. Perform some steps related to the "Deploy" stage.

SonarQube

- Plataforma Open Source para *continuous inspection*
- Fornece ferramentas e métricas visando qualidade de código
- Reviews automáticos
- Code smells
- Cobertura de testes

SonarQube


Code
From your master and branches
Build
2 C
Bugs
0 A
Vulnerabilities
0 A
Code Smells
Package
Quality Gate
Passed
Production
Deploy with confidence

OBRIGADO!

Centro
Rua Formosa, 367 - 29º andar Centro, São Paulo - SP, 01049-000

Alphaville
Avenida Ipanema, 165 - Conj. 113/114 Alphaville, São Paulo - SP,06472-002

+55 (11) 3358-7700
contact@7comm.com.br

7COMm
Serviços e Soluções em TI