Introdução ao Docker

Jean Phelipe de Oliveira Lima

jpdol.eng16@uea.edu.br

/jpdol

Introdução

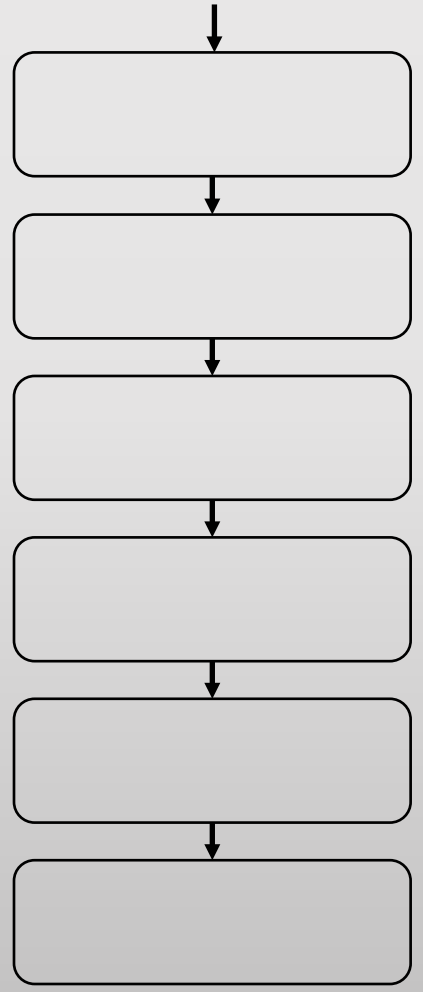
- Engenharia de Software (Revisão)
- Fase de Implantação
- Docker
- Container vs VM
- Namespaces
- Cgroup
- Union File System (UnionFS)

Engenharia de Software (Revisão)

- Fluxo de Referência para o desenvolvimento de Software.

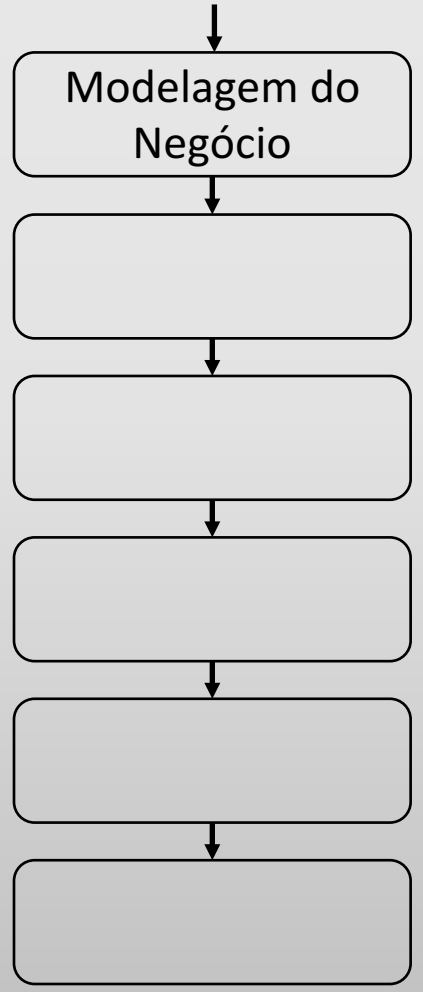
Engenharia de Software (Revisão)

- Fluxo de Referência para o desenvolvimento de Software.



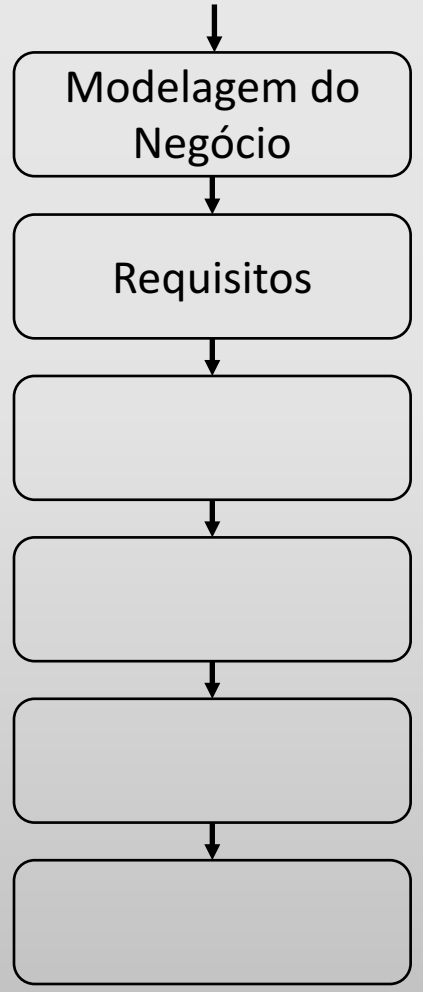
Engenharia de Software (Revisão)

- Fluxo de Referência para o desenvolvimento de Software.



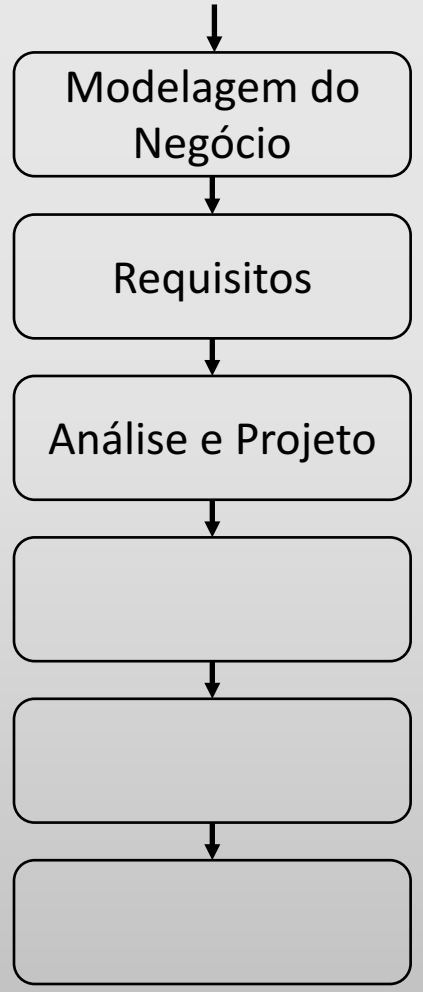
Engenharia de Software (Revisão)

- Fluxo de Referência para o desenvolvimento de Software.



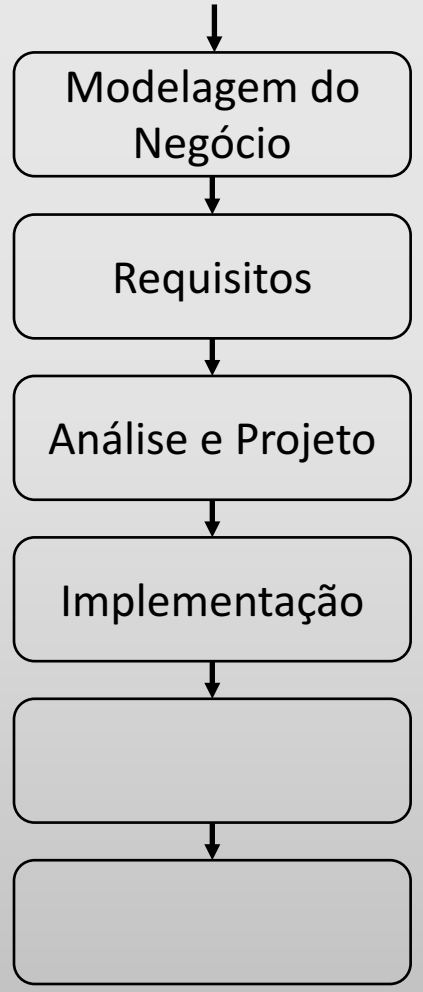
Engenharia de Software (Revisão)

- Fluxo de Referência para o desenvolvimento de Software.



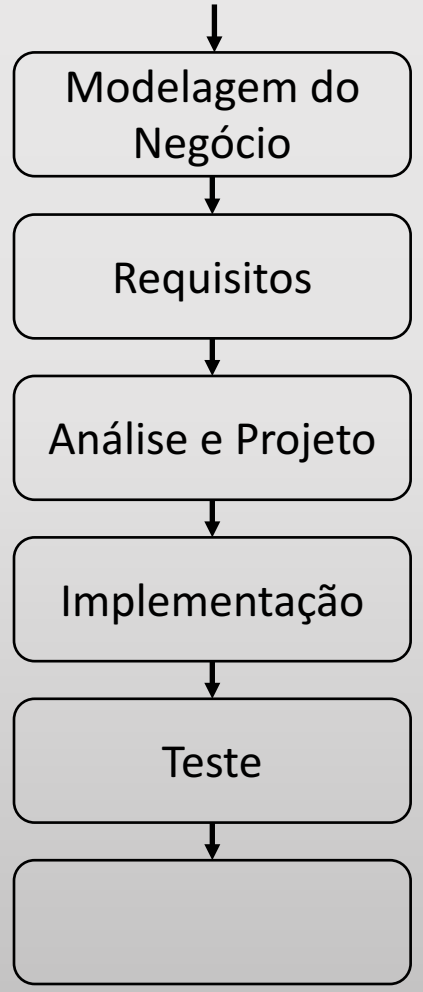
Engenharia de Software (Revisão)

- Fluxo de Referência para o desenvolvimento de Software.



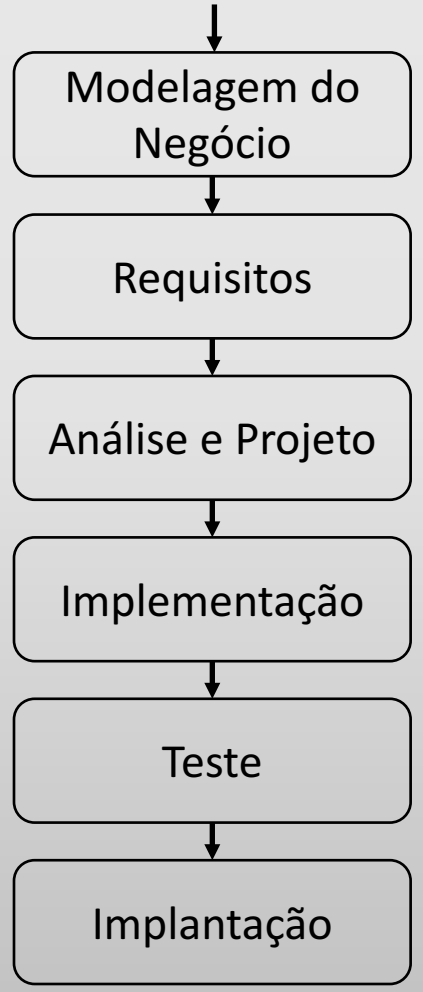
Engenharia de Software (Revisão)

- Fluxo de Referência para o desenvolvimento de Software.



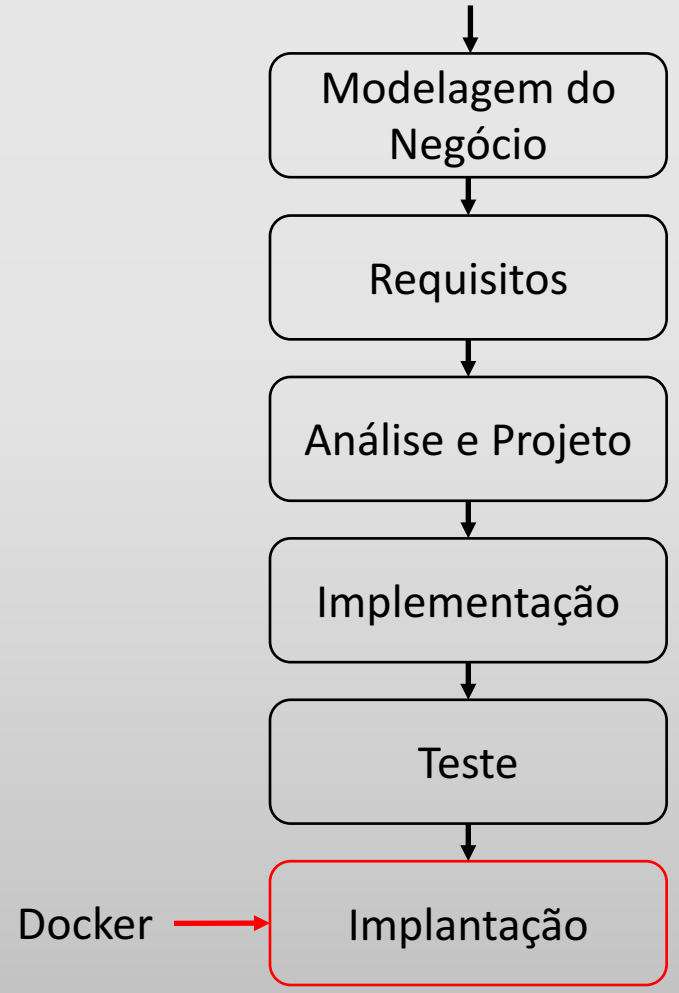
Engenharia de Software (Revisão)

- Fluxo de Referência para o desenvolvimento de Software.



Engenharia de Software (Revisão)

- Fluxo de Referência para o desenvolvimento de Software.



Fase de Implantação

- Desafios:
  - Vários Processos:
    - instalação e configuração dos servidores;
    - ajustes na aplicação para ser disponibilizada por um *web server*;
    - estabelecer comunicação com o banco de dados;
    - etc.
  - Aplicação Distribuída:
    - *Front-end;*
    - *Back-end;*
    - BD;
    - API;
    - Múltiplos Servidores;
    - etc.

Fase de Implantação

- Solução:
  - Isolar e manter a mesma configuração de cada serviço, ou camada da nossa aplicação, em diferentes ambientes.

Docker

- Ferramenta para criar e manter *containers*
  - Responsável por armazenar vários serviços de forma isolada do *SO host*
    - Ex: *web server*, banco de dados, aplicação, *memcached,* etc.
- Possui *back-end* baseado em LXC (Linux Containers)

Container vs VM

- Tipos de virtualização:
  - *Bare Metal*:
    - *Software* instalada sobre o *hardware*;
    - Alto isolamento;
    - Cada VM criada executa seu próprio kernel e instância do SO;
    - Ex: *Xen*, *VMware ESXi* e *Hyper-V.*
  - *Hosted*:
    - *Software* instalado sobre o SO host;
    - Ex: *VirtualBox*

Container vs VM

- Containers:
  - Menos isolado;
  - Compartilha recursos com o *kernel host*;
  - *Cgroups* → limitar e isolar o uso de CPU, memória, disco, rede, etc.
  - *Namespaces* → isolar grupos de processos

  - Resumo: *containers* são mais leves, já que não precisam de um ambiente virtual completo, pois o *kernel* do *host* proporciona total gerenciamento de memória, I/O, CPU, etc.

Namespaces

- Docker tira proveito do recurso de *Namespaces* para prover um espaço de trabalho isolado para os *containers*
- *Namespaces* cria uma camada de isolamento para grupos de processos.
- Quando um *container* é criado, automaticamente um conjunto de *namespaces* também é criado para ele:
  - **PID** – isolamento de processos;
  - **NET** – controle de interfaces de rede;
  - **IPC** – controle dos recursos de IPC (*InterProcess Communication*);
  - **MNT** – gestão de pontos de montagem;
  - **UTS** – isolar recursos do kernel;

Cgroups

- O segredo para executar aplicações de forma isolada é liberar apenas os recursos necessários.

- O *Cgroups* permite que o Docker compartilhe os recursos de *hardware* existentes no *host* com os *containers* e, caso seja necessário, pode definir limites de uso e algumas restrições.

Union File System (UnionFS)

- São sistemas de arquivos que funcionam por meio da criação de camadas;
- São rápidos e leves;

- O Docker utiliza sistemas de arquivos em camadas para a construção de imagens que serão usadas na criação de *containers*.

Introdução ao Docker

Jean Phelipe de Oliveira Lima

jpdol.eng16@uea.edu.br

/jpdol