**1. Bash e Docker**

**Fácil:**

* **Criar um script Bash simples** para automatizar a criação de um contêiner Docker para uma aplicação simples (exemplo: uma aplicação Node.js ou Python).
* **Subir uma aplicação Dockerizada usando Docker Compose**, criando contêineres para diferentes serviços (exemplo: backend e banco de dados).

**Médio:**

* **Criar um script Bash para atualizar** uma aplicação Dockerizada (por exemplo, atualizar a versão de uma imagem e reiniciar o contêiner automaticamente).
* **Utilizar Docker e Docker Compose para criar uma aplicação de múltiplos serviços** (frontend, backend e banco de dados) e configurar comunicação entre os serviços dentro de contêineres.

**Difícil:**

* **Desenvolver um script Bash que automatize todo o processo de provisionamento de ambiente** com Docker: desde a criação de redes, volumes e contêineres até o monitoramento básico do status dos contêineres.
* **Integrar Docker e Bash em um pipeline CI/CD** para construir, testar e implantar a aplicação automaticamente em contêineres.

**2. Visualização de Dados**

**Fácil:**

* **Criar um gráfico simples com dados estáticos** usando uma biblioteca como Chart.js ou D3.js, e exibi-lo em uma página HTML.
* **Conectar uma aplicação backend simples** (como uma API REST em Python ou Node.js) a um front-end e exibir dados de um arquivo JSON como gráfico.

**Médio:**

* **Consumir dados de uma API externa** (por exemplo, uma API pública de clima ou finanças) e exibir esses dados em um gráfico dinâmico.
* **Criar uma aplicação web que exibe uma tabela interativa com dados** e fornece funcionalidades de ordenação e filtro, usando HTML, CSS e JavaScript.

**Difícil:**

* **Criar um dashboard interativo com múltiplos gráficos** utilizando dados de uma API (por exemplo, gráficos de barras, linhas, etc.), com funcionalidades de atualização dinâmica e filtragem em tempo real.
* **Implementar uma funcionalidade de visualização de dados em tempo real**, onde o dashboard atualiza automaticamente com novos dados (por exemplo, utilizando WebSockets ou polling de API).

**3. Provisionamento de Ambientes**

**Fácil:**

* **Usar Docker Compose para provisionar um ambiente básico** com uma aplicação web e um banco de dados.
* **Criar um script simples em Bash ou Ansible** para provisionar um ambiente de desenvolvimento com ferramentas como Git, Docker e Nginx.

**Médio:**

* **Automatizar o provisionamento de um ambiente de múltiplos contêineres** usando Docker Compose e configurar variáveis de ambiente e redes.
* **Provisionar um ambiente de servidor na nuvem** (por exemplo, AWS ou Azure) e configurar Nginx como servidor web, usando Ansible.

**Difícil:**

* **Desenvolver um playbook Ansible** para provisionar e configurar uma aplicação completa em múltiplos servidores, incluindo balanceamento de carga, configuração do Nginx e instalação de dependências.
* **Configurar e automatizar o provisionamento de uma infraestrutura em nuvem com Terraform** e Ansible para implementar um ambiente escalável e resiliente.

**4. Mastering Containerization (Domínio de Containerização)**

**Fácil:**

* **Criar um contêiner Docker simples** para uma aplicação básica (por exemplo, uma aplicação em Node.js ou Python).
* **Executar e acessar um contêiner Docker em modo interativo**, e executar comandos dentro dele para verificar a configuração.

**Médio:**

* **Construir e otimizar uma aplicação Dockerizada com múltiplos serviços** (por exemplo, frontend, backend e banco de dados) utilizando Docker Compose.
* **Criar um arquivo Dockerfile** com instruções para otimizar a construção da imagem, como cache de dependências e minimização do tamanho da imagem.

**Difícil:**

* **Criar uma aplicação de microserviços usando Docker** e configurar a comunicação entre os contêineres usando redes e volumes compartilhados.
* **Automatizar a criação e o deploy de contêineres com Docker e Ansible** para garantir que a infraestrutura seja rapidamente configurável e escalável.

**5. Kubernetes**

**Fácil:**

* **Criar um cluster Kubernetes local** usando Minikube e rodar uma aplicação simples (por exemplo, uma aplicação web) em pods do Kubernetes.
* **Criar um deployment no Kubernetes** para uma aplicação Dockerizada e expor o serviço via um serviço do tipo ClusterIP.

**Médio:**

* **Escalar uma aplicação no Kubernetes** (aumentando o número de pods) e testar a escalabilidade da aplicação.
* **Configurar um Ingress Controller no Kubernetes** (usando Nginx, por exemplo) para rotear tráfego para diferentes serviços dentro do cluster.

**Difícil:**

* **Criar um cluster Kubernetes na nuvem** (usando Google Kubernetes Engine, Amazon EKS, ou Azure Kubernetes Service) e configurar um pipeline CI/CD para deploy automático no cluster.
* **Configurar monitoramento e alertas no Kubernetes** usando Prometheus e Grafana para monitorar o desempenho da aplicação e dos pods.

**6. Usando o VSCode**

**Fácil:**

* **Configurar o VSCode para trabalhar com Docker**, instalando a extensão Docker e criando contêineres diretamente do editor.
* **Usar o VSCode para editar e executar scripts Bash** dentro de um contêiner Docker, facilitando o fluxo de desenvolvimento.

**Médio:**

* **Configurar o VSCode com integração Git**, realizar commits, pull requests e push diretamente do editor.
* **Usar o VSCode como terminal integrado** para rodar comandos em um contêiner Docker ou em uma instância remota de servidor.

**Difícil:**

* **Configurar o VSCode com integração com Kubernetes** para gerenciar e visualizar pods diretamente do editor.
* **Automatizar a configuração de VSCode para múltiplos desenvolvedores** em um projeto Dockerizado, criando arquivos de configuração compartilhados (como launch.json e tasks.json).

**7. Configuração do Nginx**

**Fácil:**

* **Instalar o Nginx e configurar um servidor web básico** para servir uma página estática simples.
* **Configurar o Nginx como proxy reverso** para uma aplicação Dockerizada, redirecionando o tráfego HTTP para o contêiner.

**Médio:**

* **Configurar o Nginx para balanceamento de carga**, distribuindo as requisições entre múltiplos contêineres ou servidores.
* **Criar uma configuração de Nginx para SSL/TLS**, configurando um certificado autoassinado ou gerado por Let's Encrypt para um servidor web seguro.

**Difícil:**

* **Configurar o Nginx como Ingress Controller no Kubernetes**, roteando o tráfego para múltiplos serviços dentro de um cluster Kubernetes.
* **Configurar o Nginx para alto desempenho**, incluindo otimização de cache, compressão de conteúdo e configuração de segurança avançada.

**8. Swagger para Documentação de API**

**Fácil:**

* **Criar uma API simples em Express (Node.js) ou Flask (Python)** e documentá-la com Swagger.
* **Testar a API usando Swagger UI**, garantindo que todos os endpoints e parâmetros estejam bem descritos.

**Médio:**

* **Criar uma API RESTful** com autenticação JWT e documentar os endpoints com Swagger, incluindo exemplos de respostas e erros.
* **Integrar Swagger com uma aplicação já existente**, configurando o arquivo de documentação automaticamente a partir dos endpoints da API.

**Difícil:**

* **Documentar uma API complexa** com múltiplos recursos, relacionamentos entre dados e validações de entrada, usando Swagger para gerar a documentação automaticamente.
* **Configurar a autenticação e segurança na documentação Swagger**, permitindo que a documentação interaja com uma API que exige autenticação e controle de acesso.