INSTITUTO FEDERAL DE SANTA CATARINA

MURILO BAUER

# Ferramenta para implementação de aplicações para ensino com moderação

São José - SC

Agosto/2021

# FERRAMENTA PARA IMPLEMENTAÇÃO DE APLICAÇÕES PARA ENSINO COM MODERAÇÃO

Trabalho de conclusão de curso apresentado à Coordenadoria do Curso de Engenharia de Telecomunicações do campus São José do Instituto Federal de Santa Catarina para a obtenção do diploma de Engenheiro de Telecomunicações.

Orientador: Ederson Torresini

São José - SC

Agosto/2021

Murilo Bauer
Ferramenta para implementação de aplicações para ensino com moderação/ Murilo Bauer. – São José - SC, Agosto/2021-
39 p. : il. (algumas color.) ; 30 cm.

Orientador: Ederson Torresini

Monografia (Graduação) – Instituto Federal de Santa Catarina – IFSC
Campus São José
Engenharia de Telecomunicações, Agosto/2021.

1. Nuvem. 2. Virtualização. 3. Contêiner. 4. Kubernetes.

MURILO BAUER

# FERRAMENTA PARA IMPLEMENTAÇÃO DE APLICAÇÕES PARA ENSINO COM MODERAÇÃO

Este trabalho foi julgado adequado para obtenção do título de Engenheiro de Telecomunicações, pelo Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, e aprovado na sua forma final pela comissão avaliadora abaixo indicada.

São José - SC, 5 de Agosto de 2021:

**Ederson Torresini, Dr.**
Orientador
Instituto Federal de Santa Catarina

**Professor, Dr.**
Instituto Federal de Santa Catarina

**Professor**
Instituto Federal de Santa Catarina

**Professor**
Instituto Federal de Santa Catarina

*Para realizar grandes conquistas,*
*devemos não apenas agir, mas também sonhar,*
*não apenas planejar, mas também acreditar.*
*Anatole France*

# AGRADECIMENTOS

Dedico meus sinceros agradecimentos aqueles que muito me ajudaram para concluir este trabalho. Agradeço ao meu orientador Dr. Ederson Torresini, pelo apoio que sempre disponibilizou e a forma de orientação com entusiasmo neste trabalho. Ao meu grande amigo Bruno Marcos Espindola pela parceria nos estudos durante todo o curso. Agradeço a minha família, pois me deram toda a estrutura para que me torna-se a pessoa que sou hoje. Pelo apoio e paciência durante todo o curso. Agradeço a todos aqueles que aqui não foram mencionados, mas de alguma forma contribuíram para a realização deste trabalho.

*"O otimismo é a fé em ação.*
*Nada se pode levar a efeito sem otimismo."*
*(Helen Keller)*

# RESUMO

A evolução tecnológica, principalmente na área da computação em nuvem, vem disponibilizando novas ferramentas que auxiliam no ensino aprendizagem dos alunos e professores, oferecendo a utilização de inúmeras aplicações por meio de uma conexão de internet. Estas aplicações geralmente são oferecidas em nuvem de máquinas virtuais ou de contêineres. A CTIC do IFSC/SJ vem utilizando nuvem privada de contêineres na forma de microsserviços para otimizar a infraestrutura, ter maior disponibilidade aumentando a tolerância de falhas das aplicações e aumentar a escabilidade do sistema. O uso de contêineres apresenta ser uma solução leve e rápida para a criação de laboratórios virtuais, isto é, a criação de aplicações que possuam por exemplo aplicativos como MySQL, Imunes ou Matlab para elaboração de experimentos com roteiros, disponibilizados para os estudantes via conexão de internet. A criação destas aplicações é realizada com imagens de contêineres, geralmente utilizando o sistema Kubernetes para a orquestração destas aplicações. Estas imagens de contêineres requerem personalização das suas configurações, logo o usuário que deseja criar uma aplicação necessita conhecer um pouco a estrutura de contêineres, além de editar estas configurações diretamente no arquivo modo texto da imagem do contêiner. Este trabalho propõe o desenvolvimento de uma ferramenta que permita ao usuário gerenciar aplicações para utilização em laboratórios de diversas áreas de estudo sem a necessidade de trabalhar diretamente com as imagens de contêineres, enviando para uma ferramenta de moderação que deverá verificar as configurações selecionadas antes que o laboratório entre em produção.

**Palavras-chave**: Nuvem. Virtualização. Contêiner. Kubernetes.

# ABSTRACT

The technological evolution, mainly in the area of cloud computing, has been making available new tools that help in the teaching of students and teachers, offering the use of numerous applications through an internet connection. These applications are usually offered in the cloud of virtual machines or containers. The CTIC of IFSC/SJ has been using private container cloud in the form of microservices to dry the infrastructure, have greater availability by increasing the fault tolerance of the applications and increase the system scalability. The use of containers presents itself as a light and fast solution for the creation of virtual laboratories, that is, the creation of applications that have for example *software* such as MySQL, Immunes or Matlab for the elaboration of experiments with scripts, available for the students via internet connection. The creation of these applications is done with images of containers, usually using the Kubernets system for the implementation and management of these applications. These container images require customization of their settings, so the user who wants to create an application needs to know a little about the structure of containers, and edit these settings directly in the text mode file of the container image. This work proposes the development of a tool that allows the user to manage applications for use in laboratories of several study areas, without the need to work directly with the container images, sending it to a moderator who should check the selected settings before the laboratory goes into production.

**Keywords**: Cloud. Virtualization. Container. Kubernets.

# LISTA DE ILUSTRAÇÕES

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS E SIGLAS

# SUMÁRIO

# 1 INTRODUÇÃO

A evolução tecnológica possibilitou a criação de novas aplicações utilizadas na área de ensino, inovando o ensino aprendizagem e o tornando mais eficiente e dinâmico. Novos recursos que podem ser explorados dentro da sala de aula, mas que geralmente são utilizados para o ensino a distância permitem aos alunos acessarem aplicações utilizados para a realização de experimentos ou roteiros elaborados pelos professores, até participar de salas virtuais que permitem trocas de mensagens com os professores, otimizando o tempo gastos nas tarefas escolares e necessitando apenas acesso a internet. Aplicações que são utilizadas por todas as idades, de simples jogos que colaboram no desenvolvimento motor de crianças, a aplicações que facilitam na elaboração de projetos em circuitos digitais em Field Programmable Gate Array (FPGAs) utilizados comumente em cursos de graduação, mestrado e doutorado. Estas aplicações remotas permitem a realização de experimentos de estudo em diversas áreas como redes, programação, banco de dados, química, física, entre outros. Essas aplicações auxiliam na criação de laboratórios virtuais que consistem na elaboração de experimentos definidos por um roteiro elaborados pelos professores, onde os alunos contam com as aplicações pré-configuradas e disponibilizadas via acesso a internet. Este laboratório pode consistir em utilizar a aplicação Imunes para a criação de uma rede de computadores para analisar sua qualidade com diferentes protocolos, utilizar a aplicação MySQL para apresentar conceitos básicos no uso de banco de dados, ou na utilização da aplicação Quartus para descrever, simular e analisar dispositivos digitais. O uso destas aplicações em laboratórios virtuais podem ser acessadas de qualquer local, o aluno pode realizar de sua residência os experimentos com as aplicações e replicar fisicamente estes experimentos em sala de aula, como realizar a montagem de um circuito digital.

Com a evolução na área de rede de computadores surgiu a computação em nuvem que permitiu o fornecimento de serviços de computação como armazenamento, banco de dados, permite a utilização de outros sistemas operacionais, atualizações das aplicações disponibilizadas são realizadas de forma automática sem a intervenção do utilizador e disponibiliza aplicações antes executados de forma local sendo executadas de forma remota pelos usuários. A computação em nuvem se tornou uma grande aliada na área de ensino, pois diminui o custo com a manutenção de infraestrutura física nos institutos educacionais, e ampliou informações e recursos acessíveis de qualquer lugar, independente do tipo de dispositivo (smartphone, tablet, notebook), necessitando apenas de acesso à internet. Permite a criatividade dos professores para integrar vídeos, jogos ou aplicações para a realização de experimentos ou tarefas, auxiliando a aprendizagem que pode ir além de aulas tradicionais e livros. Segundo Almeida (2007), a utilização das tecnologias no processo educativo proporciona novos ambientes de ensinar e aprender diferentes dos ambientes tradicionais, e as reais contribuições das tecnologias para a educação surgem a medida que são utilizadas como mediadoras para a construção do conhecimento.

A virtualização foi uma das tecnologias que impulsionou esta utilização de aplicações ou recursos através da computação em nuvem. Ao virtualizar os recursos de hardware, vários ambientes podem ser criados a partir de um único hardware físico (REDHAT, 2020b). Isto permite que em um único computador ou servidor sejam geradas diversas máquinas virtuais, reduzindo os custos com a aquisição e manutenção de infraestrutura, além de melhorar a escabilidade. Outro ponto é a facilidade na migração de um ambiente computacional sem a existência de conflitos de *hardware*. O surgindo novas tecnologias como os contêineres permitiu o desenvolvimento de aplicações baseadas em microsserviços. O uso de contêineres apresenta ser uma solução leve e rápida, diferente das máquinas virtuais que necessitam de um sistema operacional para executar aplicações, os contêineres rodam em ambiente isolado utilizando o sistema operacional da máquina física, independente do *hardware* e disponibilizando recursos básicos para o funcionamento de

uma aplicação.

A arquitetura de microsserviços estabelece a necessidade de uma grande quantidade de contêineres sendo necessário um mecanismo para o controle destas aplicações, chamado de orquestrador. Uma das ferramentas conhecidas é o Kubernetes, originalmente projetado pelo Google, a ferramenta é *open-source* utilizada para automatizar a implantação, dimensionamento e a gestão das aplicações em contêineres.

No IFSC/SJ são disponibilizadas aplicações como o *Matlab*, *Quartus* de forma remota aos alunos e professores. A Coordenadoria de Tecnologia da Informação e Comunicação (CTIC) do IFSC/SJ vem utilizando nuvem privada de contêineres para disponibilizar microsserviços com o objetivo de tornar o uso dos recursos mais eficiente no atendimento de uma grande número de alunos sem a necessidade de uma grande infraestrutura, assim como garantir maior disponibilidade aumentando a tolerância de falhas das aplicações. A CTIC recebe solicitações dos professores para a disponibilização de aplicações e recursos via nuvem privada aos alunos, utilizados para a elaboração dos laboratórios virtuais que consistem na realização de um roteiro com experimentos definidos pelo professor. Atualmente o professor não possui acesso a uma ferramenta que auxilie na criação destas aplicações, este processo é realizado via solicitação por e-mail, que gera uma nova tarefa na fila da CTIC, e neste e-mail o professor necessita especificar os recursos e pacotes que serão utilizados no laboratório virtual. Em muitas ocasiões existe a falta de informações por parte do professor, sendo necessário novas trocas de mensagens, tornando a realização desta solicitação demorada. Os recursos necessários são alocados e disponibilizados aos professores, que muitas vezes não informam uma data final de utilização, tornando necessário que a CTIC questione após semanas se há possibilidade de liberar estes recursos para a utilização em novas aplicações. Além disto, na grande maioria dos casos não é informado o tamanho real do recurso necessário, implicando na alocação de recursos de forma demasiada. Com a solicitação a CTIC do IFSC/SJ para disponibilizar a aplicação solicitada manipula de forma direta as imagens dos contêineres, que consiste na criação e edição das configurações diretamente em arquivos no estilo texto, utilizando o Kubernetes para a gestão destas imagens, alocando uma parcela de tempo do profissional para esta tarefa.

## 1.1 Objetivos

Desenvolver uma ferramente que permita ao usuário (professor, servidor) gerenciar as aplicações de diversas áreas de estudo, sem a necessidade de realizar a solicitação por e-mail. Verificando a coerência das configurações selecionadas (número de processadores, uso de espaço em disco) e automatizando a criação das imagens de contêineres sem a necessidade de trabalhar diretamente nestes arquivos. Imagens de contêineres e a configuração da aplicação (portas, persistência em disco, entre outros) que serão enviadas para moderação por parte da CTIC do IFSC/SJ. Exigir o tempo de uso dos recursos, para que sejam liberados após este tempo. Criar um fluxo de solicitação, análise e aprovação para disponibilizar ambientes virtuais sobre contêineres para ensino.

## 1.2 Motivação

A motivação deste documento foi a necessidade da elaboração para aprovação e aplicar os conhecimentos adquiridos ao longo do curso de Engenharia de Telecomunicações no Instituto Federal de Santa Catarina (IFSC).

## 1.3 Organização do texto

O texto está organizado da seguinte forma: No Capítulo 2 é apresentado a fundamentação teórica. No Capítulo 3 são apresentados as ferramentas e metodologias utilizadas. Por fim, são apresentadas as

conclusões sobre este trabalho.

# 2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os fundamentos teóricos que embasam o desenvolvimento do trabalho. As seções abordam os seguintes temas: Computação em nuvem, Virtualização, Máquinas virtuais, Contêineres e Orquestração de Contêineres com Kubernetes.

## 2.1 Computação em Nuvem

A computação em nuvem (do inglês *cloud computing*) é o fornecimento de serviços de computação, incluindo servidores, armazenamento, bancos de dados, rede, *software*, análise e inteligência, pela Internet para oferecer inovações mais rápidas, recursos flexíveis e economias de escala (AZURE, 2020). Possibilita o acesso e manipulação de arquivos, o uso de diferentes aplicativos ou ferramentas de qualquer lugar pela rede internet, sem a necessidade de instalar diversas aplicações no seu computador.

A computação em nuvem colabora com a redução dos custos com a implantação e a manutenção na infraestrutura, melhorando a escabilidade pois permite a alocação de serviços ou recursos conforme necessidade.

### 2.1.1 Modelos de Serviço

Os serviços oferecidos pela computação em nuvem se enquadram em categorias amplas conforme a Figura 1, que podem ser chamadas de "pilha"de computação em nuvem, pois uma serve de camada para as outras.

- Infrastructure as a Service (IaaS) (Infraestrutura como serviço) - camada que serve como base para as demais. Conhecida como a camada que fornece os componentes básicos da infraestrutura virtualizados: processamento, armazenamento e rede. Inclui o serviço de Data Center sendo oferecido pelo provedor, responsável pelo gerenciamento de energia, resfriamento. Esta camada disponibiliza servidores, máquinas virtuais, redes e armazenamento ao usuário através de uma interface em nuvem.

- Plataform as a Service (PaaS) (Plataforma como serviço) - camada onde o provedor da nuvem disponibiliza os recursos de infraestrutura. Esta camada que oferece um ambiente para desenvolvimento de aplicações. O usuário não controla a infraestrutura, desta forma a definição de recursos não é disponível, mas possui um ambiente com acesso a banco de dados e ferramentas necessárias que facilitam aos desenvolvedores a criação de aplicações móveis ou web.

- Software as a Service (SaaS) (*Software* como serviço) - camada que distribui ao usuário final toda a funcionalidade de uma aplicação através de uma interface pela nuvem. O usuário final apenas utiliza a aplicação sem a necessidade de pensar sobre a infraestrutura ou como o serviço é mantido. Como exemplo, temos o Google Docs.

### 2.1.2 Modelos de Implantação

Há quatro diferentes modelos para implantar serviços em nuvem: nuvem privada, nuvem pública, nuvem comunitária e nuvem híbrida. Esta escolha depende dos serviços que serão implementados, a redução de custo desejada e a elasticidade.


Figura 1 – Camadas da Computação em nuvem (ZHANG LU CHENG, 2010)

Uma nuvem privada se refere aos recursos de computação em nuvem usados exclusivamente por uma única empresa ou organização. Uma nuvem privada pode estar localizada fisicamente no *datacenter* local da empresa. Algumas empresas também pagam provedores de serviços terceirizados para hospedar sua nuvem privada. Uma nuvem privada é aquela em que os serviços e a infraestrutura são mantidos em uma rede privada (AZURE, 2020), assim permitem acesso pleno a todos os recursos, que podem ser livremente alocados. Possibilita o total controle da implementação do componentes básicos da própria nuvem.

As nuvens públicas pertencem a um provedor de serviço de nuvem terceirizado e são administradas por ele, que fornece recursos de computação (tais como servidores e armazenamento) pela internet. Com uma nuvem pública, todo o hardware, software e outras infraestruturas de suporte são de propriedade e gerenciadas pelo provedor de nuvem (AZURE, 2020). A nuvem pública, em geral, é a opção com menor custo monetário, pois é pago pelos recursos que são utilizados. A infraestrutura é oferecida por uma empresa de *cloud computing* é compartilhada entre diversos clientes e toda a interação se dá por meio da internet (OLHARDIGITAL, 2020).

A nuvem comunitária é derivada da nuvem privada, sendo um conjunto de nuvens privadas compartilhadas por várias organizações/empresas e oferece suporte a uma comunidade específica que possui as mesmas preocupações (por exemplo, missão, requisitos de segurança, política e considerações de conformidade). Este tipo de modelo de implantação pode existir localmente ou remotamente e pode ser detida, gerenciada e operada por uma ou mais organizações pertencentes à comunidade, por terceiros ou alguma combinação destes (TELECO, 2020).

A nuvem híbrida é a junção de dois ou mais sistemas diferentes de nuvem para que o usuário aproveite vantagens que não teria com a adoção de um só. Muitas organizações usam o modelo híbrido para mover dados e aplicativos alternadamente nos modos público e privado. Dessa forma, ganham flexibilidade e diferentes opções de implantação, permitindo um melhor aproveitamentos dos serviços de cada nuvem. Oferece uma redução dos custos com a possibilidade de escalar quando necessário potencial adicional na nuvem pública.

## 2.2 Virtualização - Máquinas Virtuais

A virtualização permite a distribuição dos recursos físicos entre diversos ambientes ou serviços, que consiste basicamente em abstrair o *hardware*. Permite instalar e utilizar diferentes sistemas operacionais em um único servidor físico, conforme a Figura 2. Este processo é muito utilizado por desenvolvedores de *software* para simulações de compatibilidade de aplicativos em diversos sistemas operacionais. Esta tenologia possibilitou a redução de custos associados aquisição, manutenção e refrigeração.

Programas de software chamados hipervisores separam os recursos físicos dos ambientes virtuais que precisam utilizar tais recursos. Os hipervisores podem ser executados em um sistema operacional (como em um laptop) ou instalados diretamente no hardware (como um servidor), que é o tipo de virtualização preferido da maioria das empresas. Os hipervisores dividem os recursos físicos para que sejam utilizados por diferentes ambientes virtuais (REDHAT, 2020e). Bons exemplos de hipervisores são o Virtualbox, VMWare e Hyper-V. A máquina virtual é um arquivo digital que pode ser migrado e aberto em outro servidor que possua a mínima compatibilidade de arquitetura e do conjunto de instruções, precisando apenas do hipervisor para a execução.

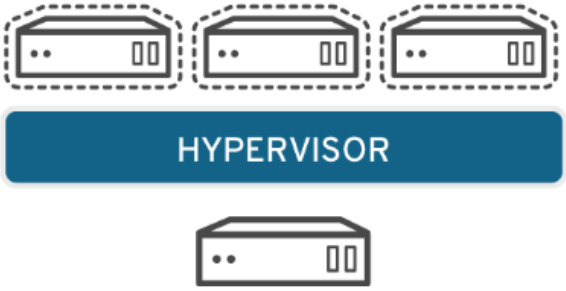


Figura 2 – Virtualização de um servidor físico (REDHAT, 2020e)

## 2.3 Contêineres

O contêiner é um conjunto de isolamento do sistema operacional em um determinado processo, através de limitação de processamento e *namespaces*, trabalhando na aplicação e suas dependências. Realiza o diálogo com o *kernel* através das chamadas de sistemas, diminuindo a alocação de recursos para um sistema virtual, pois seu objetivo não é fornecer uma máquina virtual com um sistema operacional isolado.

As máquinas virtuais utilizam um hipervisor para emular ou abstrair o *hardware*, isolando e executando um sistema operacional por inteiro. Os contêineres necessitam de menos recursos como disco, memória e processador. Os contêineres não tem visão da máquina inteira, compartilham o mesmo kernel do sistema operacional e isolam os processos da aplicação do restante do sistema (REDHAT, 2020d).

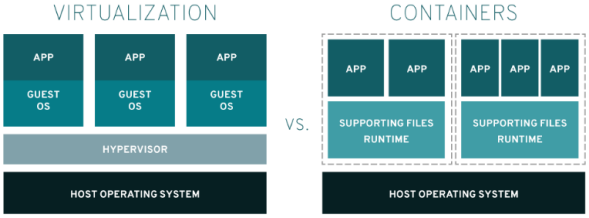


Figura 3 – Virtualização vs Containers (REDHAT, 2020d)

Contêineres podem ser criados para executar aplicações ou tarefas específicas e buscando facilitar a implantação, redimensionamento e o gerenciamento de aplicações em contêineres foram criadas ferramentas de orquestração. O Kubernetes é uma destas ferramentas de orquestração, esta é código aberto e foi criada pela empresa Google.

## 2.4 Kubernetes

O Kubernetes é uma plataforma de código aberto utilizada para orquestrar e gerenciar *clusters* de contêineres. É um ambiente integrado para criar, manter, monitorar e destruir contêineres que operam em mesmo conjunto (*pool*) de recursos de processamento, armazenamento e rede.

Segundo (REDHAT, 2020c) o Kubernetes possibilita:

- Orquestrar contêineres em vários *hosts*.
- Aproveitar melhor o hardware para maximizar os recursos necessários na execução das aplicações corporativas.
- Controlar e automatizar as implantações e atualizações de aplicações.
- Montar e adicionar armazenamento para executar aplicações com monitoramento de estado.
- Escalar rapidamente as aplicações em contêineres e recursos relacionados.
- Gerenciar serviços de forma declarativa, garantindo que as aplicações sejam executadas e mantidas em um determinado estado.
- Verificar a integridade e autorrecuperação das aplicações com posicionamento, reinício, replicação e escalonamento automáticos.

O Kubernetes utiliza o modelo *cluster*, usando APIs (*Application Programming Interface*) para descrição do estado desejado. Detalhando melhor, o usuário ao definir as imagens de container, o número de réplicas, aplicações para execução e outras informações, está definindo o estado desejado do *cluster*, esta definição é realizada através da criação dos objetos do *Kubernetes Control Plane* utilizando a API do Kubernetes. O *Kubernetes Control Plane* possui processos que são executados no *cluster*, que são subdivididos em componentes *Master* e *Node* conforme Figura 4. O componente *Master* gerencia o *cluster*, detectando e respondendo eventos.

Os componentes Kubernetes *Master* são:

- **etcd:** banco de dados distribuído do tipo chave-valor para armazenar os dados internos do *cluster* Kubernetes e de estado dos seus objetos.
- **API Server:** Expõe a API do Kubernetes e oferece uma interface para administração do *cluster*. É o principal ponto de gerenciamento do *cluster*, processa operações REST, valida e atualiza os objetos correspondentes no etcd.
- **Controller Manager:** observa o estado do *cluster* para que os recursos necessários sejam atendidos. Incorpora os objetos Controladores fornecidos pelo Kubernetes para observar a execução das aplicações de forma correta.
- **Scheduler:** aloca o pod recém criado em um *node* considerado ideal para sua execução, de forma a igualar a demanda com base nos recursos disponíveis.

Segundo (REDHAT, 2020a), os componentes Kubernetes *Node* são executados em cada nó para manter os pods em execução e fornecer o ambiente de execução dos Kubernetes.

- **kubelet:** agente que garante a execução dos containeres em um pod.
- **kube-proxy:** mantém as regras de rede dos nós, atendendo demandas internas e externas e encaminhando para o pod correto.
- **Container Runtime:** software responsável por executar os contêineres.

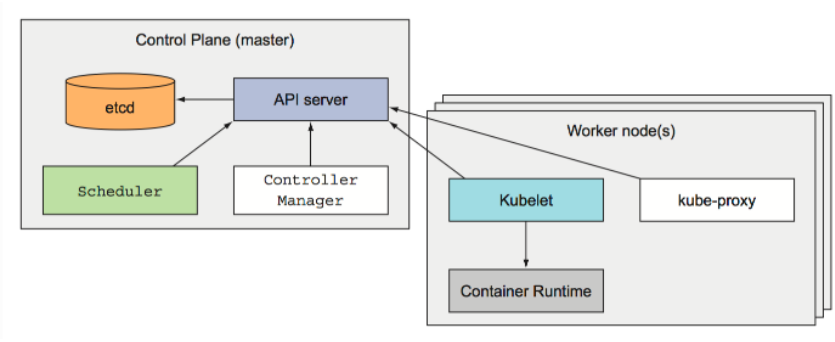


Figura 4 – Componentes de um *cluster* Kubernetes (TSUI, 2020)

O kubernetes utiliza uma série de objetos, chamados *Kubernets Objects*, para representar o estado do sistema de forma abstrata. Estes objetos disponibilizam informações como os recursos de rede e disco, aplicativos e cargas de trabalho sendo utilizadas pelo *cluster*.

Alguns dos objetos básicos podem ser definidos como:

- **Pods:** Caracterizado com um processo em execução no *cluster*. Grupo de um ou mais contêineres de aplicação implantados em um único nó. Compartilham o mesmo endereço de rede, nome de host e outros recursos.
- **Service:** disponibiliza o pod aos usuários, criando uma ponte virtual através de uma porta de um contêiner.
- **Volume:** compartilha arquivos entre os contêineres executados em um pod, e realiza a manutenção dos arquivos em caso de falha de um contêiner.
- **Namespace:** implementa a execução de múltiplos *clusters* virtuais em um único *cluster* físico. Os pods e implantações, são agrupados logicamente em um *namespace* para dividir um cluster físico e restringir a criação, a exibição ou o gerenciamento de acesso para os recursos. Podem ser criados *namespaces* para separar grupos de teste e produção. Os usuários podem interagir apenas com recursos dentro de seus*namespaces* atribuídos.

O Kubernetes fornece abstrações em um nível superior, os chamados Controladores que observam o estado atual do *cluster*, com a função de desenvolver, e até mesmo fornecer funcionalidades ou recursos adicionais para os objetos básicos.

- **ReplicaSet:** garante a quantidade de pods em execução especificada pelo usuário.
- **Deployment:** fornece a implementação e atualização de bibliotecas e versão das aplicações para os pods. Cria e atualiza o objeto ReplicaSet.

- **StatefulSet:** gerencia aplicações com estado, em caso de problemas tenta recuperar, executando conforme o estado anterior.
- **DaemonSet:** garante que todo nó possua pelo menos um pod.
- **Job:** definido como um pod que atende uma tarefa específica e deixa de existir.
- **CronJob**: Derivado do *job*, executa tarefas em intervalos específicos. Automatizando backups, tarefas de limpeza, entre outros.

## 2.5 Controle de versão

Um sistema de controle de versão tem como o objetivo gerenciar e manipular diferentes versões ou revisões de um arquivo em uma árvore de objetos. Estes arquivos ficam em um repositório criado no servidor com o sistema de controle de versão.

Assim pode ser utilizado para manter um histórico e o usuário responsável por cada modificação de especificação nos arquivos utilizados para definir o estado dos objetos em um uso no *cluster* Kubernetes, estes arquivos seguem o formato JavaScript Object Notation (JSON) ou Human-Readable Data-Serialization Languagen (YAML).

Segundo (DIAS, 2011) o controle de versão permite:

- Registro do Histórico: permite acompanhar a evolução do projeto, cada alteração e usuário responsável. Além disso, permite reconstruir uma revisão específica do arquivo sempre que desejado;
- Colaboração Concorrente: possibilita que vários desenvolvedores trabalhem em paralelo sobre o repositório, e sequencial em cada arquivo para gerar uma cadeia histórica;
- Variações no Projeto: Mantém linhas diferentes de evolução do mesmo projeto. Por exemplo, mantendo uma versão do estado corrente do objeto no Kubernetes enquanto a equipe prepara novas especificações do objeto Kubernetes.

O usuário consegue realizar uma cópia local, trabalhar nos arquivos e enviar o arquivo ao repositório com as alterações realizadas. O Sistema de Controle de Versão alerta no caso do arquivo estar desatualizados, enviando as novas informações adicionadas e permitindo mesclar as diferentes versões.

Os sistemas de controle de versão são classificados em dois tipos: Centralizados e distribuídos.

### 2.5.1 Sistema de controle de versão centralizado

Segundo (DIAS, 2011), o controle de versão centralizado segue a topologia em estrela, havendo apenas um único repositório central mas várias cópias de trabalho, uma para cada desenvolvedor. A comunicação entre uma área de trabalho e outra passa obrigatoriamente pelo repositório central.

### 2.5.2 Sistema de controle de versão distribuído

Ainda segundo (DIAS, 2011), são vários repositórios autônomos e independentes, um para cada desenvolvedor. Cada repositório possui uma área de trabalho acoplada e as operações *commit* e *update* acontecem localmente entre os dois.

Neste sistema cada usuário possui um repositório independente, cópia do original, possuindo todas as versões ou revisões dos arquivos, fazendo com que o trabalho fique independente do servidor e o trabalho continue mesmo se o servidor parar de funcionar.

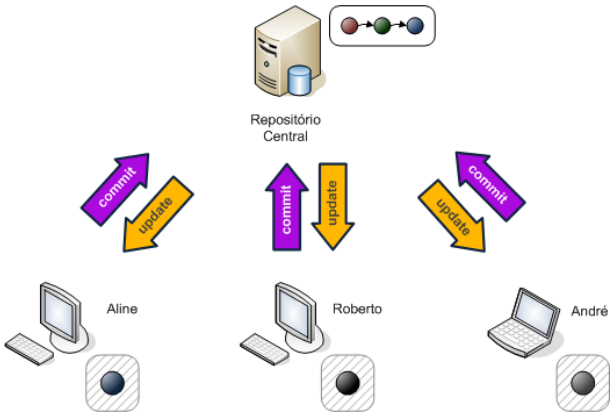


Figura 5 – Sistema de controle de versão centralizado (DIAS, 2011)

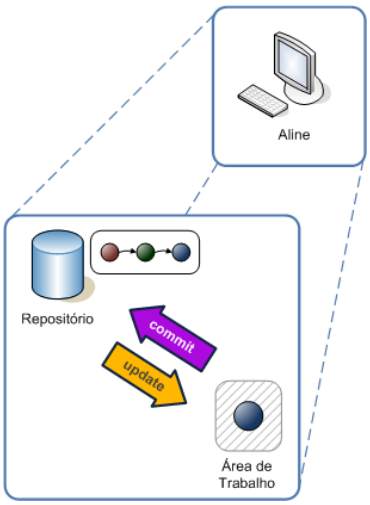


Figura 6 – Sistema de controle de versão distribuído (DIAS, 2011)

Git é um exemplo de sistema de controle de versão distribuído de código aberto e gratuito. Possibilita realizar que o mesmo arquivo seja alterado ao mesmo momento por dois usuários, salvando ambas alterações sem sobrescrever qualquer informação ou código.

## 2.6 Integração e entrega contínuas

A integração e entrega contínua CI/Continuos Delivery (CD) visa agilizar, aumentar a qualidade e segurança nas alterações do projeto.

Utilizando o comando *commit*, disponível no Git, o usuário envia o código ou arquivo com as novas especificações do objeto Kubernetes, para o repositório remoto do sistema de versão que pode ser configurado com o ciclo do CI/CD.

Segundo (REDHAT, 2021) com a integração contínua (CI), os desenvolvedores consolidam as mudanças no código de volta a uma ramificação compartilhada com mais frequência (às vezes, até diariamente). As mudanças são consolidadas e depois validadas através da criação automática da aplicação. Vários testes automatizados, geralmente de unidade e integração, são feitos para garantir que as mudanças não corrompam a aplicação. Basicamente, tudo é testado, incluindo classes, funções e diferentes módulos que formam toda a aplicação.

Ainda segundo (REDHAT, 2021), depois de realizar a automação de compilações e da unidade e os testes de integração na CI, a entrega contínua automatiza o lançamento desse código validado em um repositório. A etapa final é a implantação contínua que é um complemento da entrega contínua responsável por automatizar o lançamento de uma aplicação para a produção.

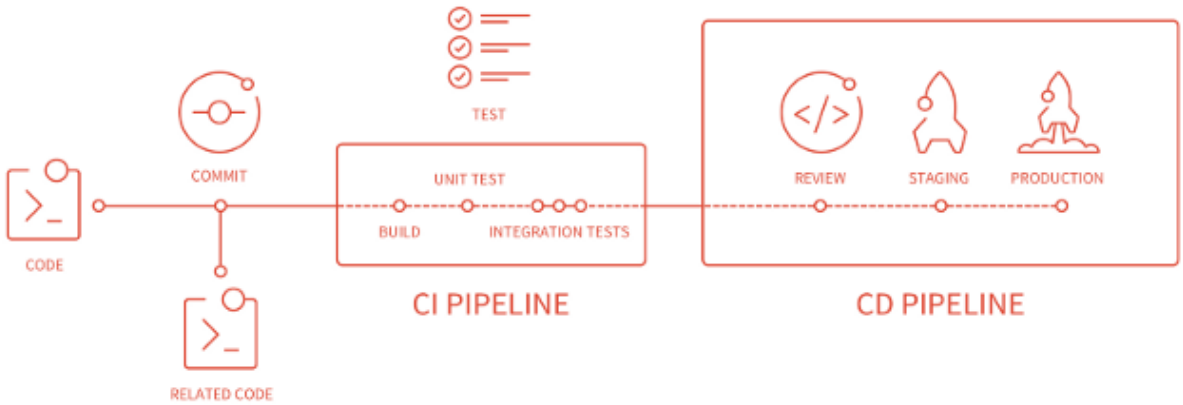


Figura 7 – Integração e Entrega Contínua (FARADAY, 2019)

## 2.7 Revisão de código

A revisão de código é uma ferramenta utilizada para garantir a qualidade de *software*, onde diversos colaboradores, diferente do autor, verificam a descrição e mudanças no código, analisando e propondo alterações no código.

A revisão de código evita que um código ou arquivo que define o estado dos objetos em um uso no *cluster* Kubernetes, seja enviado direto para a produção apresentando erros ou possíveis vulnerabilidades.

Segundo (MILANESIO, 2013), ao adotar a integração contínua (CI), seu objetivo é colocar o código em seu controle de versão o mais rápido possível para que possa ser verificado e validado com todo o projeto: em troca, você obtém um status vermelho-amarelo-verde de sua mudança. Sempre que uma construção não é verde, ela é definida como "quebrada" e a prioridade da equipe é restabelecer uma versão principal "saudável" da ramificação principal, corrigindo a construção.

Na Figura 8 é demonstrado a integração contínua sem a revisão de código. Nesta situação toda alteração de código é enviada diretamente para a ramificação principal (*branch master*) . No momento que uma compilação falha (P1) a equipe necessita intervir para consertar, sendo necessário um novo envio (P2) para restabelecer a *branch master*, sendo mantido estes registros de P1 e de P2 em seu histórico.

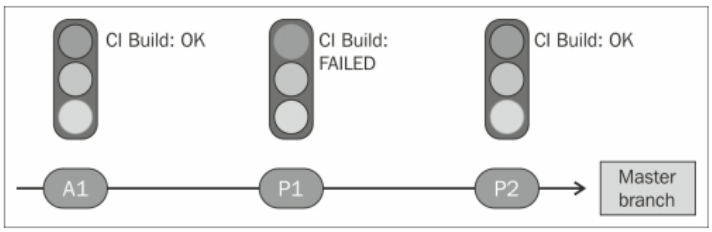


Figura 8 – Integração Contínua(CI) sem Revisão de código (MILANESIO, 2013)

Na Figura 9 é demonstrado a integração contínua com a revisão de código. Nesta situação toda nova alteração de código é enviada para uma ramificação isolada de revisão, sem influenciar no fluxo normal. Se uma nova versão (P1) possua defeitos apenas a ramificação de revisão sofre falha na compilação e assim que enviar outra versão (P2) com as correções e sem defeitos, esta será mesclada a ramificação principal.

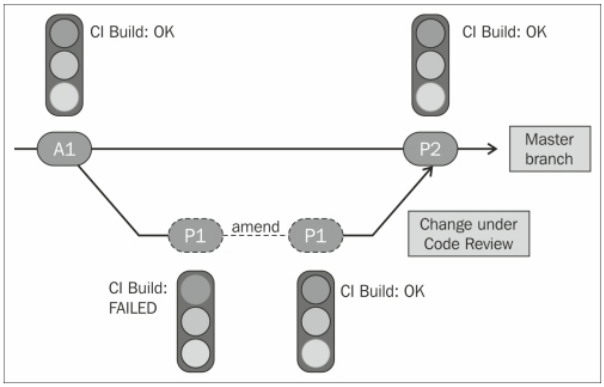


Figura 9 – Integração Contínua(CI) com Revisão de código (MILANESIO, 2013)

Neste caso a estabilidade do master não é afetada, e a ramificação de revisão permite a validação e melhoria no código antes do envio para produção.

A revisão de código possui o esquema de votação ou pontuação para aprovar ou reprovar a versão do código antes de enviar para produção. Um dos pontos é realizado de forma automática pela ferramenta de revisão de código utilizada. Além disto o código recebe a revisão e *feedback* dos revisores, como pode ou o que necessita ser melhorado no código, por fim escolhem uma pontuação para o código. Se o código for aprovado, de acordo com sua pontuação, é enviado para a produção.

Outros benefícios com a revisão de código são:

- Melhor qualidade do código: permite melhorar a capacidade de manutenção no código (legibilidade, compreensão).
- Compartilhamento de conhecimento: permite que os revisores e autores desenvolvam novas abordagens de soluções.

# 3 PROPOSTA

Neste trabalho é proposto o desenvolvimento de uma ferramenta que permita ao usuário gerenciar aplicações para utilização em laboratórios de diversas áreas de estudo sem a necessidade de trabalhar diretamente com as imagens de contêineres. Elaborando uma interface Web para professores e a equipe de TI do IFSC, através de um formulário que permita a seleção de um serviço disponível, parâmetros para o serviço (memória, quantidade de processadores, espaço em disco), data inicial e final de utilização do serviço, este último de grande importância para desalocar os recursos que não serão mais utilizados na infraestrutura do IFSC - São José.

Finalizando a seleção do serviço pela interface Web, o arquivo de descrição do objeto contêiner será elaborado/criado e enviado para uma ferramenta de moderação com controle de versão, contando com verificação em duas etapas. A primeira verificação é automatizada, realizada pela própria ferramenta de moderação, enquanto a segunda verificação é por revisores definidos, neste caso a equipe CTIC do IFSC que devem revisar o arquivo de descrição. Esta moderação tem como objetivo validar se o arquivo possui algum erro de sintaxe ou erros de configurações, como um exemplo de erro simples temos o caso do usuário selecionar o uso de 80gb ao invés de 8gb de memória para a utilização do laboratório. Permitindo que os revisores façam as correções, se necessário.

A ferramenta de moderação enviará o laboratório para produção utilizando o sistema Kubernetes para a orquestração destas aplicações utilizadas, mediante revisão e a aprovação na ferramenta de moderação. Por fim, enviar via e-mail ao usuário a confirmação das aplicações do laboratório em produção com as informações de acesso como IP público, porta, *login* e senha.

### 3.0.1 Plano de trabalho

No plano de trabalho esta inserido o cronograma do TCC2. As principais etapas que foram realizadas durante o TCC1 e as que serão feitas no TCC2, são as seguintes:

1. Estudo e implementação de interface Web com formulário com opções de configuração das aplicações;
2. Estudo e implementação dos arquivos de descrição dos contêineres e orquestração com Kubernets.
3. Estudo e implementação de controle de versão com ferramenta de moderação;
4. Implementação do envio das informações para os usuários. Demais testes e validações das ferramentas.
5. Escrita do documento do TCC ( Implementação, Análises, Resultados e Conclusões), entrega deste para a banca, preparação da apresentação, defesa, correção do documento, e entrega da versão final para a coordenação do curso;

A Tabela 1 apresenta o cronograma de execução das futuras etapas.

| Etapas | 2021 | | | | |
|---|---|---|---|---|---|
| | Mai | Jun | Jul | Ago | Set |
| 1 | X | | | | |
| 2 | X | X | | | |
| 3 | | X | X | | |
| 4 | | | X | X | |
| 5 | | | | X | X |

Tabela 1 – Cronograma do Projeto.

# REFERÊNCIAS


ALMEIDA, M. E. B. de. *TECNOLOGIAS DIGITAIS NA EDUCAÇÃO: O FUTURO É HOJE*. 2007. Disponível em: <https://etic2008.files.wordpress.com/2008/11/pucspmariaelizabeth.pdf>. Citado na página 23.

AZURE, M. *O que é computação em nuvem?* 2020. Disponível em: <https://azure.microsoft.com/pt-br/overview/what-is-cloud-computing/>. Citado 2 vezes nas páginas 27 e 28.

DIAS, A. F. *Conceitos Básicos de Controle de Versão de Software — Centralizado e Distribuído*. 2011. Disponível em: <https://blog.pronus.io/posts/controle-de-versao/conceitos-basicos-de-controle-de-versao-de-software-centralizado-e-distribuido/>. Citado 3 vezes nas páginas 15, 32 e 33.

FARADAY, G. *O que é CI/CD? Onde eu uso isso?* 2019. Disponível em: <https://gabriel-faraday.medium.com/o-que-e-ci-cd-onde-eu-uso-isso-57e9b8ad8c73>. Citado 2 vezes nas páginas 15 e 34.

MILANESIO, L. *Learning Gerrit Code Review*. 2013. Disponível em: <https://subscription.packtpub.com/book/application_development/9781783289479/1/ch01lvl1sec08/benefits-of-code-review>. Citado 3 vezes nas páginas 15, 34 e 35.

OLHARDIGITAL. *Nuvem pública, privada e híbrida: entenda as diferenças*. 2020. Disponível em: <https://olhardigital.com.br/bluemix/tipos_de_nuvens.php>. Citado na página 28.

REDHAT. *Componentes do Kubernetes*. 2020. Disponível em: <https://kubernetes.io/docs/concepts/overview/components/>. Citado na página 31.

REDHAT. *Introdução à virtualização*. 2020. Disponível em: <https://www.redhat.com/pt-br/topics/virtualization>. Citado na página 23.

REDHAT. *Kubernetes e a tecnologia de containers*. 2020. Disponível em: <https://www.redhat.com/pt-br/topics/containers/what-is-kubernetes>. Citado na página 30.

REDHAT. *O que é um container Linux?* 2020. Disponível em: <https://www.redhat.com/pt-br/topics/containers/whats-a-linux-container>. Citado 2 vezes nas páginas 15 e 29.

REDHAT. *Virtualização: o que é, como funciona e quais os seus benefícios*. 2020. Disponível em: <https://www.redhat.com/pt-br/topics/virtualization/what-is-virtualization>. Citado 2 vezes nas páginas 15 e 29.

REDHAT. *Integração e entrega contínuas: pipeline CI/CD*. 2021. Disponível em: <https://www.redhat.com/pt-br/topics/devops/what-is-ci-cd>. Citado 2 vezes nas páginas 33 e 34.

TELECO. *Serviços em Nuvem I: Modelos de Implantação*. 2020. Disponível em: <https://www.teleco.com.br/tutoriais/tutorialservnuvopers1/pagina_4.asp>. Citado na página 28.

TSUI, C. *Architecture of a Kubernetes cluster*. 2020. Disponível em: <https://carltsuis-blog.readthedocs.io/en/latest/kubernetes/components-of-k8s.png>. Citado 2 vezes nas páginas 15 e 31.

ZHANG LU CHENG, R. B. Q. *Cloud computing: state-of-the-art and research challenges*. 2010. Disponível em: <https://link.springer.com/content/pdf/10.1007/s13174-010-0007-6.pdf>. Citado 2 vezes nas páginas 15 e 28.