Autologic: uma solução para o provisionamento automatizado do Weblogic

Nicolas G. Teixeira[1] Edgar Noda[2]

[1]Instituto Federal de Educação, Ciência e Tecnologia – Campus Hortolândia (IFSP)

[1]Análise e Desenvolvimento de Sistemas

nicolas.teixeira@aluno.ifsp.edu.br

Abstract. *This paper presents the development of an automated pipeline for provisioning cluster and standalone environments on the Oracle Weblogic platform, utilizing Ansible, GitLab, and Jenkins. The proposed automation aims to replace the manual provisioning process, which is error-prone and time-consuming, with a more efficient and reliable approach. Ansible is used for automated configuration, GitLab for version control, and Jenkins for continuous integration. While this approach offers significant improvements in efficiency and error reduction, it faces challenges such as complex initial setup and ongoing maintenance requirements. The proposed solution promises to optimize provisioning time and enhance operational efficiency for companies using the platform.*

Resumo. *Este artigo apresenta o desenvolvimento de uma pipeline automatizada para o provisionamento de ambientes clusters e standalone na plataforma Oracle Weblogic, utilizando Ansible, GitLab e Jenkins. A automação proposta visa substituir o provisionamento manual, que é propenso a erros e consome muito tempo, com um processo mais eficiente e confiável. Ansible é usado para a configuração automatizada, GitLab para o controle de versão e Jenkins para a integração contínua. Embora essa abordagem ofereça melhorias significativas na eficiência e na redução de erros, enfrenta desafios como a configuração inicial complexa e a necessidade de manutenção contínua. A solução proposta promete otimizar o tempo de provisionamento e aumentar a eficiência operacional das empresas que utilizam a plataforma.*

1. Introdução e Motivação

Este Trabalho de Conclusão de Curso (TCC) baseia-se em um caso de uso real de uma empresa, no qual enfrentam desafios relacionados ao provisionamento de ambientes de TI. A necessidade de otimizar processos manuais e melhorar a eficiência operacional foi o ponto de partida para o desenvolvimento de uma solução automatizada. Antes de entrar em detalhes sobre o caso de uso e as motivações que levaram à criação desta solução, é importante definir alguns conceitos fundamentais que embasam este trabalho: automação e provisionamento.

A automação refere-se ao uso de tecnologias e ferramentas para executar processos repetitivos de forma automática, sem a necessidade de intervenção humana. No contexto da TI, a automação é uma peça-chave para melhorar a produtividade, reduzir erros e garantir a consistência das operações (SOMMERVILLE, 2011).

Dentro do escopo deste trabalho, o conceito de provisionamento é central. O provisionamento é o processo de configuração e preparação de sistemas de TI, como servidores, redes e aplicativos, para que estejam prontos para uso (SMITH, 2020). Esse

processo pode ocorrer de três maneiras: manual, semi-automatizada e automatizada.

O provisionamento manual é realizado diretamente por administradores de sistemas, que executam cada etapa do processo de forma individual, como a instalação de software e a configuração de servidores. Embora permita maior controle sobre cada etapa, esse método é extremamente suscetível a erros humanos e pode ser demorado, especialmente em ambientes complexos. Estudos mostram que o provisionamento manual em grandes infraestruturas pode levar horas e, em alguns casos, dias para ser concluído (PARETO, 2019).

Já o provisionamento semi-automatizado é uma abordagem híbrida, onde algumas etapas do processo são automatizadas, mas ainda requerem intervenção humana em momentos críticos. Essa abordagem é útil em cenários onde ajustes manuais são necessários ou quando a automação completa não é viável devido a especificidades do ambiente. Embora mais eficiente que o provisionamento manual, o processo semi-automatizado ainda pode ser limitado em termos de escalabilidade (KANG et al., 2014).

Por fim, o provisionamento automatizado utiliza ferramentas que realizam todo o processo sem necessidade de intervenção humana, permitindo a configuração de ambientes de forma rápida, padronizada e escalável. Com essa abordagem, o tempo de provisionamento é drasticamente reduzido, e a consistência entre os ambientes é garantida, minimizando os riscos de erros e retrabalho (ANSIBLE, 2022). Esse tipo de provisionamento é especialmente vantajoso em empresas que precisam gerenciar múltiplos servidores simultaneamente, aumentando a eficiência e reduzindo os custos operacionais.

Após compreender o conceito de provisionamento, é importante abordar o Oracle WebLogic Server, sistema utilizado pela empresa, para contextualizar o estudo de caso. O Oracle WebLogic Server é uma plataforma de servidor de aplicações Java EE que oferece uma infraestrutura robusta para o desenvolvimento, implantação e gerenciamento de aplicações empresariais. Ele é projetado para suportar aplicativos complexos, oferecendo recursos avançados, como gerenciamento de transações, segurança, e integração com outros sistemas e tecnologias. Com sua capacidade de fornecer um ambiente estável e escalável, o WebLogic é amplamente utilizado em grandes empresas para executar aplicações críticas, o que destaca a importância de uma abordagem eficiente e eficaz para seu provisionamento (WEBLOGIC, 2008).

O Oracle WebLogic Server pode ser configurado de duas maneiras principais: *standalone* e *cluster*. No modo *standalone*, um único servidor WebLogic opera de forma autônoma, adequado para ambientes de desenvolvimento ou aplicações de menor escala. Esta configuração permite que o servidor gerencie todas as funções sem a necessidade de coordenação com outros servidores. Por outro lado, o modo cluster envolve a utilização de múltiplos servidores WebLogic trabalhando em conjunto para formar um grupo coeso. Esta configuração é ideal para ambientes de produção que requerem alta disponibilidade e escalabilidade, pois permite o balanceamento de carga e garante a continuidade do serviço em caso de falhas. O clustering também proporciona uma gestão centralizada e uma maior capacidade de resposta a demandas elevadas, o que é crucial para aplicações empresariais que necessitam de robustez e desempenho constante (WEBLOGIC, 2008).

Agora que introduzimos os conceitos fundamentais de provisionamento e o que é o Oracle WebLogic Server, vamos aprofundar o estudo de caso, que se concentra no provisionamento do mesmo. A motivação para este trabalho surge da necessidade crítica de

melhorar este processo específico. Na empresa onde o estudo foi realizado, o provisionamento do Oracle WebLogic Server é realizado de forma semi-automatizada, utilizando dois scripts que funcionam com interface gráfica onde o analista faz interações durante o processo. Esse processo será detalhado mais a fundo no tópico 3. Aplicações Semelhantes, onde está como os scripts auxiliam o analista e quais são as limitações e vantagens dessa abordagem semi-automatizada.

No estudo de caso, esse processo semi-automatizado exige entre 4 e 6 horas de trabalho de um Analista, variando conforme sua experiência e a complexidade do ambiente a ser configurado e pela elevada interação humana, causa inconsistências e falta de padronização nos ambientes provisionados, aumentando o risco de erros e a necessidade de retrabalho.

Durante o estudo de caso, foi identificado que a empresa possui um total de 200 servidores com o Oracle WebLogic Server instalado. Desses, 132 servidores estão operando com versões desatualizadas do software. Essas versões obsoletas apresentam problemas de segurança e não oferecem as funcionalidades mais recentes disponíveis na versão atualizada do WebLogic. A desatualização representava um risco significativo para a segurança e a eficiência operacional, além de limitar o acesso a novos recursos e melhorias oferecidas pela versão mais recente do software.

Diante da necessidade de atualizar esses servidores, a empresa solicitou um orçamento para uma consultoria externa, que, em 2022, estimou um custo de R$150.000,00 e uma duração de 3 a 4 meses para a realização da atividade. Visto isso o orçamento e o prazo, este TCC propõe uma solução para resolver o estudo de caso de forma mais eficaz, com um processo de provisionamento automatizado que visa otimizar os custos e o tempo necessário para a atualização dos servidores, garantindo uma solução mais eficiente e com maior controle sobre a padronização e a segurança dos ambientes provisionados.

2. Fundamentação Teórica

A automação na área de Tecnologia da Informação (TI) vem se destacando como um fator essencial para otimizar processos e garantir maior eficiência operacional. Segundo Sommerville (2011), a automação em TI tem como objetivo minimizar a intervenção humana em tarefas repetitivas, permitindo que os profissionais se concentrem em atividades estratégicas e mais complexas. Isso é particularmente importante em ambientes de infraestrutura, onde a configuração manual de sistemas pode ser propensa a erros e demorada. O uso de ferramentas automatizadas para execução de tarefas previamente manuais garante maior padronização e consistência nos resultados.

Processos complexos, como provisionamento de servidores, demandam grande esforço e conhecimento técnico, o que pode ser simplificado por meio da automação. Kang, Lee e Koh (2014) destacam que a automação de processos complexos encapsula a complexidade por trás de uma interface simples, facilitando o controle de tarefas que anteriormente exigiam altos níveis de especialização. No caso do provisionamento de servidores de aplicação, como o Oracle WebLogic, a automação reduz drasticamente o tempo necessário para configurar ambientes complexos, além de minimizar os riscos de erros.

2.1. Automação de Infraestrutura (IaC)

A automação de infraestrutura, também conhecida como *Infrastructure as Code (IaC)*, consiste na prática de gerenciar e provisionar recursos por meio de scripts em vez de processos manuais. Smith (2020) afirma que essa abordagem garante que a infraestrutura seja configurada de maneira consistente, replicável e escalável, eliminando a necessidade de configurar manualmente cada componente de TI.

Ferramentas de gerenciamento de configuração desempenham um papel central nesse processo. O Ansible, por exemplo, permite que os administradores definam e gerenciem o estado de seus sistemas por meio de playbooks escritos em YAML, que podem automatizar desde a instalação de pacotes até a configuração completa de um servidor. Ansible é uma ferramenta amplamente utilizada por sua simplicidade e capacidade de integração com outras soluções de automação, promovendo maior controle e padronização no gerenciamento de configurações (ANSIBLE, 2022).

2.2. Versionamento de Código

O versionamento de código é a prática de registrar e controlar as mudanças em um projeto de software ao longo do tempo. De acordo com Humphrey (1997), o versionamento é crucial para garantir a rastreabilidade, permitindo que desenvolvedores acompanhem o histórico de alterações, identifiquem versões específicas do código e revertam mudanças, se necessário. Isso é fundamental em ambientes de desenvolvimento colaborativo, onde várias pessoas trabalham no mesmo projeto.

O GitLab é uma plataforma de DevOps que facilita o versionamento e o controle de código, integrando ferramentas para repositórios Git, *Continuous Integration (CI)*, *Continuous Delivery (CD)* e gerenciamento de projetos. Além de permitir a colaboração eficiente entre equipes, o GitLab oferece um histórico detalhado de todas as alterações no código, possibilitando rastrear quem fez cada modificação, quando e por quê (GITLAB, 2024).

2.3. Pipelines

No contexto de automação, uma pipeline representa uma sequência de processos automatizados que, juntos, formam um fluxo de trabalho contínuo para a compilação, teste e implementação de software. Smith (2020) explica que pipelines de automação são fundamentais para garantir que o ciclo de desenvolvimento seja contínuo e confiável, promovendo uma integração mais ágil entre as etapas de desenvolvimento e operação.

Uma pipeline não apenas automatiza o processo de provisionamento e deployment, mas também oferece uma visão clara e intuitiva de todas as fases do processo, o que facilita a identificação de possíveis erros e a manutenção do sistema. O Jenkins, por exemplo, é uma ferramenta de integração contínua que gerencia e executa pipelines automatizadas. Ele permite que os desenvolvedores visualizem cada etapa da pipeline, tornando mais fácil monitorar o progresso e solucionar problemas em tempo real (SMITH, 2020).

2.4. Gerenciamento de Projeto Ágil / Kanban

O gerenciamento ágil de projetos tem como objetivo promover a flexibilidade, colaboração e

entrega contínua de valor, adaptando-se rapidamente às mudanças e necessidades do projeto. Segundo Kang, Lee e Koh (2014), metodologias ágeis, como o Scrum e o Kanban, facilitam a gestão de projetos de desenvolvimento de software ao focar na entrega incremental e contínua de produtos.

O Kanban é uma metodologia visual que utiliza cartões para representar tarefas e colunas que indicam o status dessas tarefas (a fazer, em andamento, concluído, etc.). Sua simplicidade permite que equipes tenham uma visão clara das atividades, ajudem a identificar gargalos e promovam um fluxo contínuo de trabalho. No contexto do projeto Autologic, a abordagem Kanban será usada para organizar o desenvolvimento de funcionalidades incrementais, com foco na melhoria contínua e na entrega frequente de resultados (KANG, 2014).

3. Aplicações Semelhantes

A principal motivação por trás do desenvolvimento do Autologic reside na necessidade de uma solução Open Search que automatize completamente o provisionamento de ambientes WebLogic. Embora a Oracle fornece alguns scripts para automatizar partes desse processo, o Autologic se destaca por oferecer melhorias e funcionalidades que não estão presentes nas soluções disponíveis, proporcionando uma abordagem mais abrangente e eficiente.

Os scripts da Oracle representam uma alternativa baseada em linha de comando ao console convencional, porém, ambos os métodos, seja via script ou console, seguem uma abordagem semelhante. Isso resulta em tempos de provisionamento semelhantes e sem a efetiva mitigação de possíveis erros humanos.

Para realizar o provisionamento de ambientes utilizando métodos tradicionais, é necessário seguir os passos descritos no site da Oracle WebLogic (ORACLE, 2024):

1. É feito download do binário do Weblogic na versão a ser instalada no site da Oracle e da versão do java que será usada pelo mesmo.

2. O Analista cria um esquema de pastas no servidor destino, que será usada durante a instalação e pela aplicação.
3. Mover o arquivo para o servidor destino
    a. O movimento do arquivo é feito via ftp da máquina do analista para o servidor onde será instalado o Weblogic
4. É realizada a instalação do java.
5. Colocar as permissões corretas nos arquivos de instalação
    a. Deve-se colocar permissão de *root* nos arquivos de instalação, pois o mesmo mexe na estrutura do sistema operacional (SO).
6. O Analista executa o script que abre uma interface gráfica e fará as 8 interações que são exigidas.
    a. Durante a execução do processo, tem que interagir várias vezes, colocando informações do ambiente.
7. O analista executa um segundo script que também é uma interface gráfica e possui 8 interações, para fazer mais configurações.
8. Criar os arquivos de senha criptografada
9. Executar os scripts de start do weblogic
10. Dentro do weblogic mudar o modo de segurança para SSL

Como já foi mostrado, nesse cenário o provisionamento do WebLogic, consome, em média, de 4 a 6 horas. Esta estimativa é baseada na análise do tempo necessário para completar todas as interações e configurações manuais envolvidas, conforme demonstrado pelo processo atual.

4. Solução Proposta: Autologic

De acordo com as seções anteriores, os desafios associados ao provisionamento de ambientes WebLogic, tanto em modo *standalone* quanto em *cluster*, evidenciam a necessidade de uma solução automatizada para otimizar esse processo. Esta seção apresenta a proposta de arquitetura e ferramentas do Autologic, uma solução projetada para automatizar o provisionamento desses ambientes, abordando as dificuldades identificadas e aprimorando a eficiência operacional.

Como um dos objetivos do projeto é a otimização de tempo e redução de custos, foram selecionadas tecnologias *open source*, que além de serem gratuitas, oferecem benefícios como:

- Confiabilidade e Acesso Aberto: O *open source* proporciona confiança e transparência, permitindo a visualização e otimização do código, o que resulta em maior controle sobre as funcionalidades e correções, sem depender exclusivamente de um fornecedor para atualizações. Ao contrário dos *softwares* proprietários, onde cada modificação pode implicar custos adicionais, o *open source* permite adaptações de forma gratuita. Segundo Raymond (1999), a natureza do *open source* permite uma colaboração contínua e uma melhoria constante do *software*, tornando-o mais robusto e confiável ao longo do tempo.
- Ampla Rede de Suporte: Ampla Rede de Suporte: *Softwares open sources* contam com comunidades robustas. Quanto maior o número de usuários engajados, maior é o pool de conhecimento disponível para solucionar problemas e aprimorar funcionalidades. Essa diversidade de perspectivas oferece um suporte vasto, com debates, sugestões e auxílio para resolver questões ou melhorar a utilização do *software*. Segundo von Hippel (2005), as comunidades são um recurso valioso que promove a inovação e a resolução de problemas de forma colaborativa, aproveitando a contribuição de uma ampla base de usuários e desenvolvedores.
- Flexibilidade e Colaboração: Projetos *open sources*, como o Linux, exemplificam o poder do desenvolvimento colaborativo. Milhares de programadores ao redor do mundo contribuem com melhorias, correções de *bugs* e novos recursos. Essa colaboração global resulta em sistemas robustos, seguros e constantemente atualizados, sem depender exclusivamente de uma única equipe de desenvolvedores. Como destaca Raymond (1999), o modelo de desenvolvimento aberto do Linux não apenas acelera a inovação, mas também melhora a qualidade e a segurança do *software* por meio da contribuição coletiva e da revisão contínua.

Diante dos benefícios identificados, as ferramentas selecionadas para o Autologic foram escolhidas com base em suas funcionalidades específicas e em requisitos não funcionais, como a gratuidade e a adequação ao contexto do projeto.

O Ansible foi escolhido por sua principal vantagem de não exigir a instalação de um

agente no servidor de destino. Ele utiliza conexões SSH, necessitando apenas de uma relação de confiança entre os servidores ou das credenciais apropriadas. Embora o Jenkins e o GitLab também atendam aos requisitos, seu principal diferencial é que são ferramentas já homologadas na empresa do estudo de caso.

As Figuras 1 e 2 ilustram o funcionamento do Autologic nos diferentes cenários de provisionamento. No ambiente *standalone* (Figura 1), o processo começa com um *commit* no GitLab, que aciona um *job* no Jenkins. Este *job*, por sua vez, chama o Ansible para realizar a instalação do WebLogic na máquina de destino. Para ambientes em *cluster* (Figura 2), o processo é similar, mas o Ansible executa a instalação simultaneamente em múltiplas máquinas, garantindo que todas sejam configuradas de forma coordenada e eficiente. Essas representações demonstram a automação e a escalabilidade do Autologic em ambos os tipos de provisionamento.

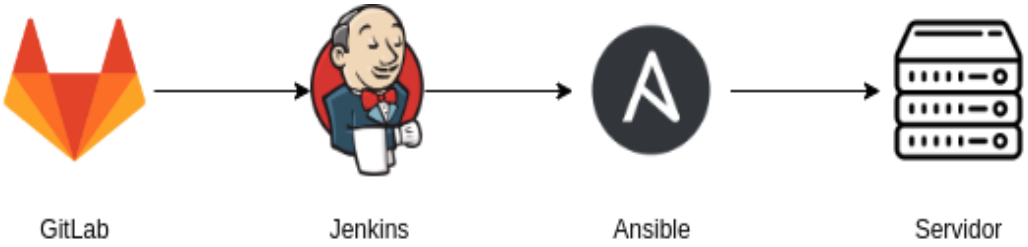

Figura 1. Cenário Autologic Standalone

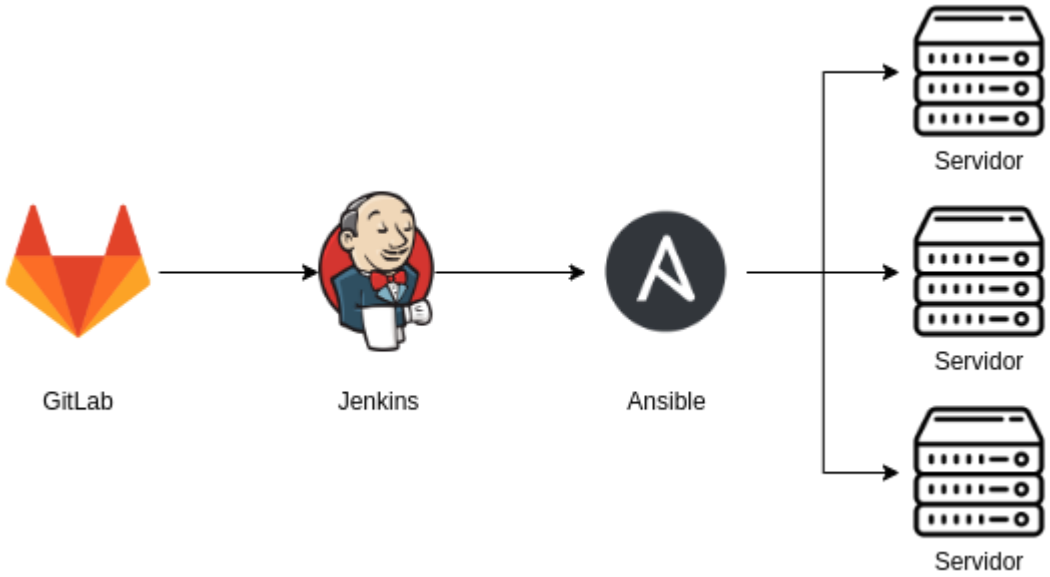

Figura 2. Cenário Autologic Cluster

5. Gestão e Planejamento

Para garantir que o Autologic seja desenvolvido de forma organizada e eficiente, foi adotada uma abordagem de entregas incrementais, utilizando um quadro Kanban. O desenvolvimento foi dividido em cinco fases principais: Planejamento da *Release*, Desenvolvimento, Testes, Entrega e *Review*. Essas fases foram aplicadas na implementação de diversas *features* fundamentais, como o estudo do processo atual de provisionamento do WebLogic, configuração do ambiente de desenvolvimento, desenvolvimento de *scripts* e *playbooks* do Ansible, entre outros.

Planejamento da *Release*: Durante essa fase, são definidas quais *features* serão incorporadas em cada release. Isso envolverá detalhar atividades macro e micro, identificando o caminho crítico para cada feature. O planejamento será refletido no cronograma, estabelecendo metas claras de entrega ao longo do tempo previsto.

Desenvolvimento: Nesta etapa, os requisitos estabelecidos no planejamento serão transformados em código funcional com ênfase nna entrega de um código de alta qualidade.

Testes: Essa fase envolverá a automação dos cenários descritos durante o desenvolvimento, garantindo que o código passe por todos os testes, incluindo testes de regressão automatizados.

Entrega: O foco será garantir uma entrega rápida e automatizada da release desenvolvida.

Review: Por fim, a fase de *review* será destinada a documentar e arquivar as lições aprendidas durante o desenvolvimento de cada *feature*.

Essa abordagem de planejamento e execução permitirá que o projeto Autologic seja entregue com alta qualidade e dentro dos prazos estabelecidos, reforçando a eficácia da arquitetura proposta.

6. Desenvolvimento

Nesta seção, detalharemos o desenvolvimento do Autologic, explicando como cada ferramenta foi utilizada para alcançar uma automação eficiente no provisionamento de ambientes Weblogic. O Autologic foi construído integrando diversas tecnologias para automatizar tarefas complexas e garantir a consistência em todas as etapas do processo de provisionamento.

Cada ferramenta desempenhou um papel essencial: o Jenkins foi utilizado para automatizar e visualizar as etapas do *pipeline* de provisionamento; o GitLab serviu para o versionamento do código e gestão colaborativa dos *scripts* e *playbooks*; e o Ansible foi empregado para orquestrar a configuração automatizada dos ambientes. A seguir, cada uma dessas ferramentas será explorada em detalhes, destacando suas configurações, integrações, e como contribuíram para o desenvolvimento eficiente do Autologic.

6.1. Jenkins

O Jenkins possui diversas *features*, porém para o Autologic foi utilizado a funcionalidade de esteira, nela temos uma visão clara de cada fase do processo (JENKINS, 2022). A Figura 3 é uma visão da esteira no Jenkins.

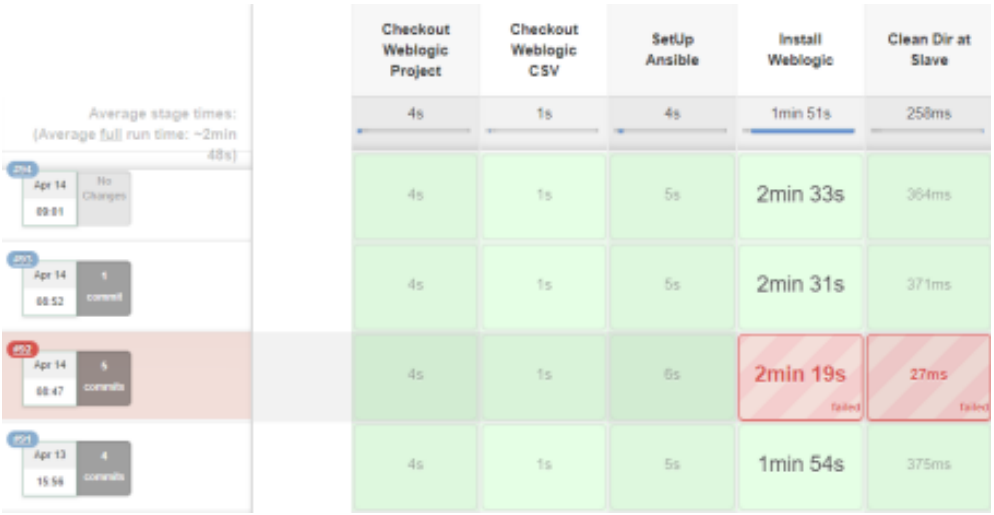

Figura 3. Visão Esteira Jenkins

Cada coluna é uma fase do processo, então o Autologic é dividido nas seguintes fases:

- Checkout Weblogic Project;
- Checkout Weblogic CSV;
- SetUp Ansible;
- Install Weblogic;
- Clean Dir at Slave

A ferramenta é visual, então se o processo for um sucesso, todas as fases do processo ficam destacadas em verde, como pode observar na Figura1 nas execuções: 91, 93 e 94. Se der algum erro, ele fica vermelho na etapa que foi o erro, como pode ser visto na execução 92. Para definir quais são e o que faz cada fase, é preciso desenvolver um arquivo texto que lembra um Json (*JavaScript Object Notation*).

A etapa Checkout Weblogic Project, faz o download da versão atual dos scripts e playbook desenvolvida em Ansible para o provisionamento do Weblogic. Na Figura 4 é a visão no *Jenkinsfile* do Checkout Weblogic Project.

A linha 13 inicia todo o processo no Jenkins, estabelecendo o nome do processo como "Checkout Weblogic Project", o qual aparecerá como a primeira coluna na execução, conforme mostrado na Figura 3, "Visão Esteira Jenkins". Em seguida, as linhas 15 a 17 especificam que esse processo será executado em uma máquina Linux pré-configurada. As linhas 21 a 29 determinam que, dentro desse processo, as ações serão realizadas na pasta scripts. A linha 24 é responsável por limpar essa pasta para evitar qualquer resíduo de execuções anteriores, garantindo um ambiente limpo. A linha 27 realiza o clone do código do Ansible que será utilizado nos passos subsequentes. Finalmente, as linhas 32 a 42 incluem mensagens para possível resolução de problemas, configuradas de acordo com

condições específicas.

```
stages {

    stage ('Checkout Weblogic Project') {

            agent {
                label "linux"
            }

            steps {

                dir("scripts") {

                    //Clear directory
                    deleteDir()

                    //Checkout dos scripts
                    git url: 'https://git.ultra.com.br/cor/weblogic/weblogic_install.git', credentialsId: '98c4be08-3d34-4ebe-aad1-4536dc71fdae'

                }
            }

            post {
                always {
                    echo 'checkout pipeline-tools post always'
                }
                success{
                    echo 'checkout pipeline-tools post success'
                }
                failure{
                    echo 'checkout pipeline-tools post failure'
                }
            }

        }

```

Figura 4. Visão Checkout Weblogic Project no Jenkinsfile.

A fase Checkout Weblogic CSV, faz o download do arquivo CSV que o usuário insere as informações necessárias para o provisionamento do Weblogic. Na Figura 5 é a visão no *Jenkinsfile* do Checkout Weblogic CSV.

O processo da linha 46 a 77 é bem similar ao já visto. Checkout Weblogic Project. Com as seguintes diferenças que (1) o processo será chamado de Checkout Weblogic CSV, (2) a pasta em que será executado é o csv e o (3) clone será feito em outro repositório, que possui um arquivo csv com informações importantes para subir o Weblogic.

```
        stage ('Checkout Weblogic CSV') {

                agent {
                    label "linux"
                }

                steps {

                    dir("csv") {

                        //Clear directory
                        deleteDir()

                        //Checkout dos scripts
                        git url: 'https://git.ultra.com.br/cor/weblogic/weblogic_install_package.git', credentialsId: '98c4be08-3d34-4ebe-aad1-4536dc71fdae'

                    }
                }

                post {
                    always {
                        echo 'checkout pipeline-tools post always'
                    }
                    success{
                        echo 'checkout pipeline-tools post success'
                    }
                    failure{
                        echo 'checkout pipeline-tools post failure'
                    }
                }

            }

```

Figura 5. Visão Checkout Weblogic Project no Jenkinsfile.

A fase SetUp Ansible executa o *script* main.sh, que interpreta o arquivo CSV do usuário e configura o *playbook* com elas. Na Figura 6 é a visão do *Jenkinsfile* do SetUp Ansible e a Figura 7 é uma imagem do código do script main.sh.

Na Figura 6, é bem similar aos outros códigos dos tópicos anteriores, com as seguintes diferenças: (1) o processo será chamado de SetUp Ansible, (2) ele será executado em uma máquina Linux e (3) será executado em uma pasta chamada scripts.

Ainda na Figura 6, da linha 89 a 91, executa um script chamado main passando como parâmetro $WORKSPACE.

Para entender o código do *script*, é importante analisar a Figura 7. A linha 1 define que o interpretador será a linguagem *bash*. Da linha 3 à 12, são declaradas variáveis essenciais para a execução do *script*. A linha 14 movimenta o conteúdo da pasta *packages* para outra pasta chamada *package*, que será utilizada mais adiante. Da linha 19 à 31, o script cria dois arrays: um com os valores das colunas do arquivo CSV e o outro com os valores correspondentes dessas colunas, e então cria variáveis com a chave do primeiro *array* e o valor correspondente do segundo *array*. Da linha 33 à 44, essas variáveis são usadas para atualizar os arquivos de configuração do Ansible. Por fim, a linha 46 movimenta os arquivos para o servidor destino onde o Weblogic será instalado.

A fase do Install Weblogic é a principal do Autologic, ele executa o *playbook* do Ansible é nele que faz o provisionamento do Weblogic. Na Figura 8 é a visão no *Jenkinsfile* do Install Weblogic.

Na Figura 8, é bem similar aos outros códigos dos tópicos anteriores, com as seguintes diferenças: (1) o processo será chamado de Install Weblogic, (2) ele será executado em uma máquina linux e (3) será executado em uma pasta chamada *scripts*.

Na linha 27, ele entra na pasta playbook, onde está o código do Ansible e na linha 28 ele executa o *playbook* para instalar o Weblogic.

Na fase Clean Dir at Slave faz a limpeza dos arquivos baixados e gerados nas fases anteriores. Na Figura 9 é a visão no Jenkinsfile do Clean Dir at Slave.

Figura 6. Visão SetUp Ansible no Jenkinsfile.

```
#!/bin/bash

# Declare Weblogic Variables
Workspace=$1
csv=$1/csv/*.csv
Scripts=$1/scripts
# Create Weblogic Pass
domain_password="ODYzZGI0ZD14"
# Create Dir Variables
host_vars_dir="${Scripts}/playbook/group_vars/all.yml"
host_install_dir="${Scripts}/install"
packages="/home/jenkins/weblogic/packages"

cp ${packages}/* ${host_install_dir}/packages/


echo $host_install_dir

#Get Values From CSV
IFS=';' read -r -a Values <<< `cat ${csv} | head -2 | tail -1`

#Get Header From CSV
IFS=';' read -r -a Header <<< `cat ${csv} | head -1`

# Get
lenght=${#Header[@]}

for (( i=0; i<$lenght; i++))
do
    echo ${Values[i]}
done

# Input Ansible variables at all.yml file
sed -i 's|${host_install_dir}|'$host_install_dir'|g' $host_vars_dir
sed -i 's|${domain_name}|'${Values[7]}'|g' $host_vars_dir
sed -i 's|${domain_password}|'$domain_password'|g' $host_vars_dir
sed -i 's|${admin_name}|'${Values[8]}'|g' $host_vars_dir
sed -i 's|${admin_address}|'${Values[5]}'|g' $host_vars_dir
sed -i 's|${server_name}|'${Values[9]}'|g' $host_vars_dir
sed -i 's|${server_address}|'${Values[0]}'|g' $host_vars_dir
sed -i 's|${cluster_name}|'${Values[10]}'|g' $host_vars_dir
sed -i 's|${product}|'${Values[3]}'|g' $host_vars_dir
sed -i 's|${product}|'${Values[3]}'|g' $host_install_dir/weblogic/wls.rsp
sed -i 's|${HOST_NAME}|'${Values[0]}'|g' ${Scripts}/playbook/hosts

scp -r ${host_install_dir}/ weblogic@${domain_name}:/apl/middleware/
```

Figura 7. Código do script main.sh.

```
stage ('Install Weblogic') {

        agent {
            label "linux"
        }

        steps {
            dir("scripts") {

                sh '''
                    cd playbook/
                    ansible-playbook provisioning.yml -i hosts
                '''
            }
        }
    }
```

Figura 8. Visão Install Weblogic no Jenkinsfile.

Figura 9. Visão Clean Dir at Slave no Jenkinsfile.

Na Figura 9, é bem similar aos outros códigos dos tópicos anteriores, com as seguintes diferenças: (1) o processo será chamado de Clean Dir at Slave, (2) ele será executado em uma máquina *linux* e (3) será executado dentro das pastas *scripts* e csv. Na linha 107 e 113, ele limpa as pastas, a fim de que na próxima execução essas pastas estejam vazias.

6.2. GitLab

O GitLab desempenha um papel crucial no projeto Autologic, sendo utilizado para o versionamento de todo o código envolvido, incluindo *scripts*, *playbooks* do Ansible e o arquivo CSV fornecido pelo usuário. As Figuras 10 e 11 ilustram a estrutura dos arquivos no GitLab. O arquivo README.md serve como uma documentação do projeto, fornecendo instruções detalhadas sobre o uso do Autologic e orientações para a instalação do Weblogic. Já o arquivo WeblogicInstall.csv é preenchido pelo usuário com as informações necessárias para que o Autologic execute o *deploy* do Weblogic de forma automatizada.

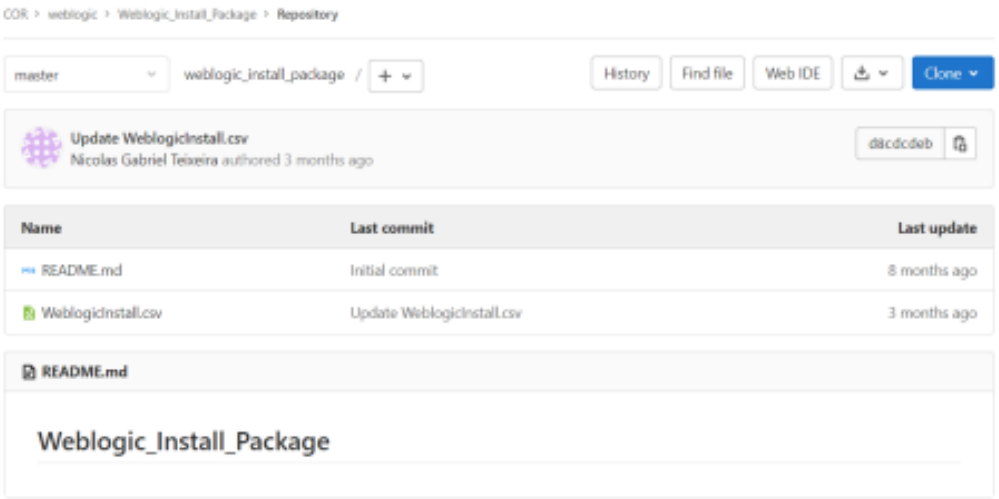

Figura 10. Visão no GitLab no arquivo CSV.

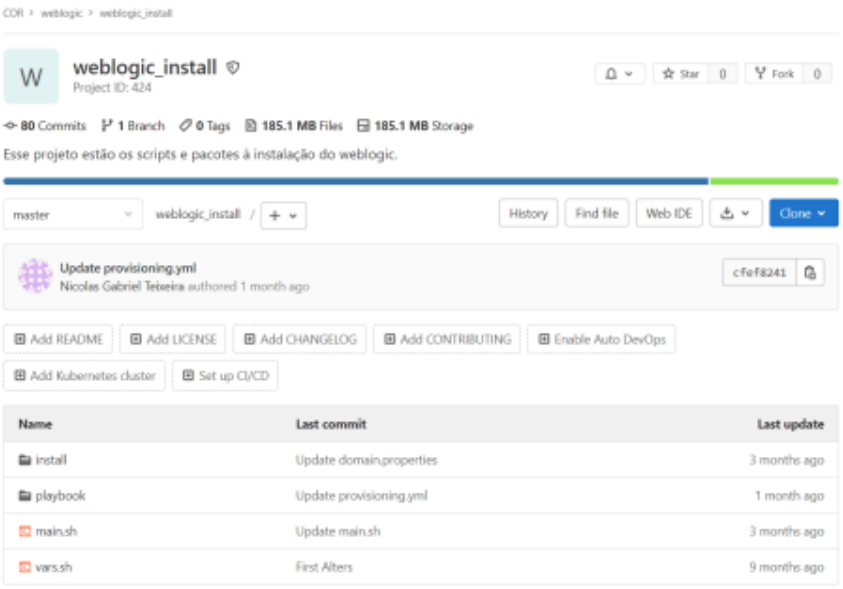

Figura 11. Visão no GitLab dos scripts e playbook.

6.3. Ansible

O Ansible é a principal ferramenta do Autologic, nela está todo o passo a passo do provisionamento de um ambiente Weblogic.

No Ansible, cada passo macro é dividido em *roles*, na Figura 12, é a visão dessa parte no projeto. Cada *role* tem um arquivo main.yml e nele é desenvolvido as etapas para que a *role* seja executada, que pode ser vista na Figura 13, da *role* weblogic.

Como é visto, neste projeto existem as roles:

- setup_install
- setup_server
- weblogic

No Ansible, é preciso de um arquivo chamado provisioning.yaml para orquestrar quais *roles* e em qual ordem será executada. Como é possível ver na Figura 13.

Na linha 2, configura qual servidor será executado as *roles*, este *host* é configurado pelo script main.sh. Das linhas 3 a 6, declara quais *roles* serão executados e em qual ordem.

Na Figura 14, é mostrado o processo de atualização do arquivo Domain.Properties, que é responsável por armazenar as informações de configuração que o Weblogic utilizará. Este arquivo é essencial para o provisionamento e configuração do ambiente Weblogic. As informações configuradas no Domain.Properties incluem:

Weblogic_Hostname: O nome do host onde o Weblogic será instalado.

Empresa: Nome da empresa que está realizando a instalação.

Ambiente: Tipo de ambiente (por exemplo, desenvolvimento, teste ou produção).

Produto: O produto específico que está sendo configurado.

Sub-Produto: Qualquer subcategoria ou módulo adicional do produto.

Admin_Address: Endereço de administração do Weblogic.

QTDE JVMs_Machine: Quantidade de JVMs (Java Virtual Machines) que a máquina deve ter.

Domain_Name: Nome do domínio a ser configurado no Weblogic.

Admin_Name: Nome do administrador do Weblogic.

Server_Name_JVM: Nome do servidor e a configuração da JVM.

Cluster_Name: Nome do cluster, se estiver configurado um ambiente de cluster.

Na linha 2, declara o nome do processo que será Update Domain.Properties. Na linha 3 funciona em conjunto com as linhas 4 a 14, a variável {{ item }}, será substituído por cada linha do with_items.

Na Figura 15, é o passo de preparar o ambiente para instalar o Weblogic. Das linhas 2 a 11, o script executa o passo Create mains Folder, que é responsável por criar os diretórios necessários, como binários e oracle, e atribuir permissões de usuário e grupo com o nome Weblogic. Em seguida, das linhas 13 a 17, o passo Copy binaries to Server é executado para copiar os arquivos de instalação para uma pasta de configuração no servidor. Entre as linhas 19 e 23, o passo Install Java instala a versão específica do Java requerida pelo Weblogic. Das linhas 25 a 28, o passo Create a Soft Link to Java cria um link simbólico para o Java, que será utilizado nos processos subsequentes de instalação. Por fim, das linhas 30 a 37, o passo Set JAVA_HOME and PATH in bash_profile configura o arquivo bash_profile para que a versão instalada do Java seja reconhecida e utilizada por todo o sistema.

Na Figura 16, é apresentado o processo de instalação do Weblogic. Das linhas 2 a 4, o script realiza o passo Unzip Weblogic jar, que descompacta o arquivo de instalação do Weblogic. Em seguida, das linhas 6 a 8, o passo Install Weblogic executa a instalação do Weblogic, deixando-o pronto para uso básico. Das linhas 10 a 14, o passo Config Domains aplica as personalizações necessárias ao Weblogic, ajustando-o com base nas configurações especificadas no arquivo domain.properties. Entre as linhas 16 e 20, o passo Create Security Folder cria um diretório para armazenar arquivos de usuário e senha utilizados pelo sistema. Das linhas 22 a 27, o passo Create Boot.Properties cria um arquivo criptografado para armazenar o usuário e senha do sistema. Das linhas 29 a 33, o passo Alter SecureListener at nodemanager.properties configura o arquivo nodemanager.properties, enquanto das linhas 35 a 39, o passo Change Type NodeManager to "Plan" ajusta o arquivo config.xml, com o objetivo de definir o protocolo de conexão padrão como HTTP. Finalmente, das linhas 41 a 43, o passo Start Weblogic Server inicia o Weblogic em segundo plano, e das linhas 45 a 47, o passo Start Node Manager inicia o Node Manager também em segundo plano, permitindo que o Weblogic esteja pronto para uso e operação.

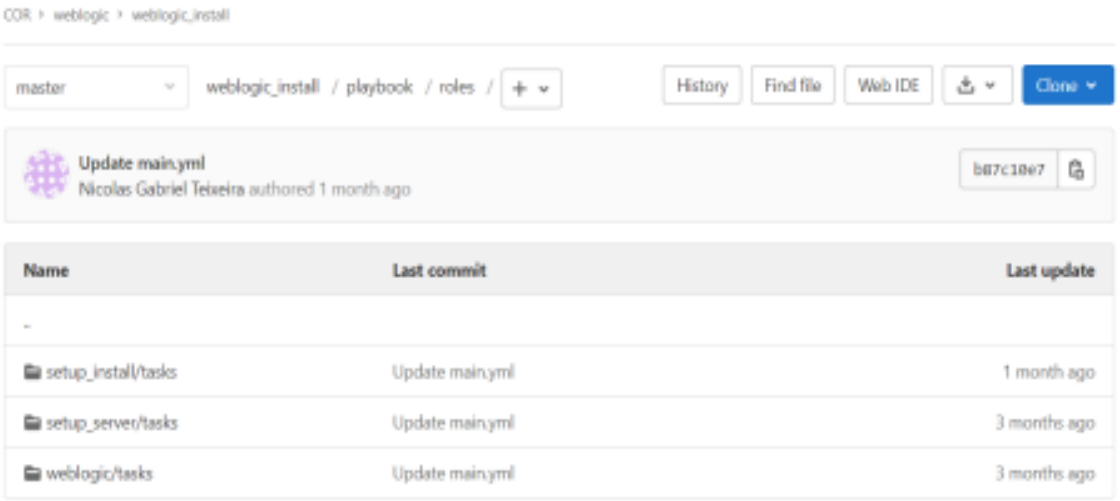

Figura 12. Roles do Projeto Autologic

```yaml
---
- hosts: weblogic
  roles:
      - setup_install
      - setup_server
      - weblogic
```

Figura 13. Código do provisioning.yml

```yaml
---
- name: Update Domain.Properties
  shell: "{{ item }}"
  with_items:
    - sed -i 's|${oracle_dir}|{{ oracle_dir }}|g' {{ install_dir }}/domain/domain.properties
    - sed -i 's|${domain_dir}|{{ domain_dir }}|g' {{ install_dir }}/domain/domain.properties
    - sed -i 's|${domain_name}|{{ domain_name }}|g' {{ install_dir }}/domain/domain.properties
    - sed -i 's|${domain_username}|{{ domain_username }}|g' {{ install_dir }}/domain/domain.properties
    - sed -i 's|${domain_password}|{{ domain_password }}|g' {{ install_dir }}/domain/domain.properties
    - sed -i 's|${admin_name}|{{ admin_name }}|g' {{ install_dir }}/domain/domain.properties
    - sed -i 's|${admin_address}|{{ admin_address }}|g' {{ install_dir }}/domain/domain.properties
    - sed -i 's|${server_name}|{{ server_name }}|g' {{ install_dir }}/domain/domain.properties
    - sed -i 's|${server_address}|{{ server_address }}|g' {{ install_dir }}/domain/domain.properties
    - sed -i 's|${cluster_name}|{{ cluster_name }}|g' {{ install_dir }}/domain/domain.properties
```

Figura 14. Código da Role setup_install

```yaml
---
- name: Create mains Folder
  file:
    path: "{{ item }}"
    state: directory
    owner: weblogic
    group: weblogic
  with_items:
    - "{{ binarios_dir }}"
    - "{{ oracle_dir }}"
#    - "{{ inventory_dir }}"

- name: Copy binarios to Server
  become_user: weblogic
  shell: "{{ item }}"
  with_items:
    - "cp {{ install_dir }}/packages/* {{ binarios_dir }}/"

- name : Install java
  become_user: weblogic
  shell: "tar -xzf {{ binarios_dir }}/jdk-*-linux-x64.tar.gz -C {{ middleware_dir }}/"
  args:
   chdir: "{{ middleware_dir }}/"

- name: Crate a Soft link to java
  become_user: weblogic
  shell: "ln -s {{ middleware_dir }}/jdk*/bin/java {{ middleware_dir }}/java"
  ignore_errors: true

- name: Set JAVA_HOME and PATH in bash.profile
  become_user: weblogic
  shell: "{{ item }}"
  with_items:
    - "sed -i '/PATH=/i JAVA_HOME={{ middleware_dir }}/java' /home/weblogic/.bash_profile"
    - "sed -i 's/PATH=/PATH=$JAVA_HOME:/' /home/weblogic/.bash_profile"
    - "sed -i '/export PATH/a export JAVA_HOME' /home/weblogic/.bash_profile"
    - "source /home/weblogic/.bash_profile"
```

Figura 15. Código da Role setup_server

```yaml
---
- name: Unzip Weblogic jar
  become_user: weblogic
  shell: "unzip {{ binarios_dir }}/fmw_* -d {{ binarios_dir }}/"

- name: Install Weblogic
  become_user: weblogic
  shell: "{{ middleware_dir }}/java -Xmx1024m -jar {{ binarios_dir }}/fmw_*.jar -silent -responseFile {{ install_dir }}/weblogic/wls.rsp -invPtrLoc {{ install_dir }}/weblogic/oraInst.loc"

- name: Config Domains
  become_user: weblogic
  shell: "{{ item }}"
  with_items:
    - ". {{ oracle_dir }}/wlserver/server/bin/setWLSEnv.sh && cd /apl/middleware/ && {{ middleware_dir }}/java weblogic.WLST {{ install_dir }}/domain/create_domain.py -p {{ install_dir }}/domain/domain.properties"

- name: Create Security Folder
  become_user: weblogic
  shell: "{{ item }}"
  with_items:
    - mkdir -p {{ domain_dir }}/{{ domain_name }}/servers/{{ admin_name }}/security

- name: Create Boot.Properties
  become_user: weblogic
  shell: "{{ item }}"
  with_items:
    - echo password={{ domain_password }} > {{ domain_dir }}/{{ domain_name }}/servers/{{ admin_name }}/security/boot.properties
    - echo username={{ domain_username }} >> {{ domain_dir }}/{{ domain_name }}/servers/{{ admin_name }}/security/boot.properties

- name: Alter SecureListener at nodemanager.properties
  become_user: weblogic
  shell: "{{ item }}"
  with_items:
    - sed -i 's/SecureListener=true/'SecureListener=false'/g' {{ domain_dir }}/{{ domain_name }}/nodemanager/nodemanager.properties

- name: Change Type NodeManager to "Plan"
  become_user: weblogic
  shell: "{{ item }}"
  with_items:
    - sed -i '/<name>{{ admin_address }}<\/name>/r {{ install_dir }}/nodeManager/config.xml' {{ domain_dir }}/{{ domain_name }}/config/config.xml

- name: Start Weblogic Server
  become_user: weblogic
  shell: "nohup {{ domain_dir }}/{{ domain_name }}/bin/startWebLogic.sh &"

- name: Start Node Manager
  become_user: weblogic
  shell: "nohup {{ domain_dir }}/{{ domain_name }}/bin/startNodeManager.sh &"
```

Figura 16. Código da Role Weblogic

7. Resultados e Discussão

Para fins de análise, o Autologic já realizou 94 processos de provisionamento, abrangendo 72 ambientes *standalones* e 22 *clusters*, totalizando 132 servidores provisionados.

A comparação entre o processo anterior e o novo sistema revelou uma melhoria significativa no tempo de provisionamento para cada categoria de ambiente do WebLogic. Anteriormente, o esforço necessário para provisionar uma máquina variava entre 4 e 6 horas. Com a implementação do Autologic, esse tempo foi reduzido drasticamente para 4 minutos para ambientes *standalones* e 10 minutos para *clusters*.

Essa redução substancial no tempo de provisionamento representa uma economia operacional considerável. A agilidade adquirida não só melhora a eficiência geral dos processos, mas também permite uma rápida resposta a novas demandas. Com o tempo economizado, a equipe pode agora se concentrar em atividades mais estratégicas e desafiadoras, aumentando a produtividade e fomentando a inovação dentro da empresa.

Além da economia de tempo, o Autologic reduziu significativamente os erros de processos manuais, que antes resultaram em inconsistências e falhas. A automação garantiu maior padronização e confiabilidade no provisionamento, minimizando a necessidade de correções e aumentando a estabilidade dos ambientes. Isso melhorou a eficiência operacional e a segurança dos processos.

O Autologic trouxe um benefício adicional ao tornar o processo de execução muito mais intuitivo através do Jenkins. Isso possibilitou que pessoas sem formação técnica também possam operar o sistema com facilidade, democratizando o acesso ao provisionamento de ambientes WebLogic.

Adicionalmente, o Autologic foi premiado em primeiro lugar em eficiência operacional em um sistema de premiação que ocorre anualmente na empresa.

8. Conclusão

A aplicação do Autologic no cenário alvo demonstrou vantagens significativas em comparação com os métodos tradicionais de provisionamento, como processos manuais e scripts fornecidos pela Oracle. A implementação de uma solução automatizada resultou em uma redução drástica no tempo de provisionamento, diminuindo o esforço necessário para cada ambiente Weblogic de horas para minutos. Essa melhoria não só acelera o processo, mas também reduz significativamente os custos operacionais associados ao provisionamento.

Além da economia de tempo e custo, a solução também contribuiu para uma redução substancial de erros. A padronização e automação proporcionadas pelo Autologic minimizam a ocorrência de falhas humanas, garantindo um provisionamento mais consistente e confiável. Isso não apenas melhora a eficiência operacional, mas também aumenta a confiabilidade dos ambientes Weblogic provisionados.

Os resultados obtidos destacam a eficácia da solução implementada e evidenciam a aplicabilidade prática dos conhecimentos adquiridos ao longo do curso de Análise e

Desenvolvimento de Sistemas. A integração de teoria e prática mostrou-se crucial para a criação de uma solução inovadora e eficiente, que responde eficazmente aos desafios reais enfrentados no ambiente corporativo. Para o desenvolvimento do Autologic, disciplinas estudadas durante o curso, como lógica de programação, qualidade de software, serviços de redes e inglês técnico avançado, foram essenciais.

Com o reconhecimento em primeiro lugar em eficiência operacional em um prêmio anual da empresa, o Autologic solidifica sua posição como uma solução eficaz, trazendo benefícios tangíveis em termos de tempo, custo e precisão. Esta experiência reforça a importância de aplicar conhecimentos acadêmicos na resolução de problemas complexos e demonstra o impacto positivo que soluções bem desenvolvidas podem ter no ambiente de trabalho.

Referências

ANSIBLE. Ansible Documentation. Disponível em: https://docs.ansible.com. Acesso em: 5 set. 2024.

GITLAB. GitLab Documentation. Disponível em: https://docs.gitlab.com. Acesso em: 5 set. 2024.

HUMPHREY, W. S. Managing the Software Process. Reading: Addison-Wesley, 1997.

JENKINS. Jenkins Automation Server. Disponível em: https://www.jenkins.io/. Acesso em: 20 nov. 2022.

KANG, H.; LEE, J.; KOH, J. Automation of Complex Processes. In: Proceedings of the International Conference on Automation. 2014.

KANG, H. Agile Project Management with Kanban. In: International Journal of Project Management. 2014.

ORACLE. WebLogic Installation Tutorial. [S. l.], [s. d.]. Disponível em: https://www.oracle.com/webfolder/technetwork/tutorials/obe/fmw/wls/12c/12_1_3/01/installwls.html. Acesso em: 07 set. 2024.

PARETO, V. Automation in IT Operations: Improving Efficiency and Reducing Costs. Journal of Information Technology, v. 12, n. 3, p. 45-58, 2019.

RAYMOND, E. S. The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. O'Reilly Media, 1999.

SMITH, J. Continuous Integration and Pipelines. In: DevOps Handbook. 1. ed. Chicago: TechBooks, 2020.

SMITH, J. Infrastructure as Code: Managing and Provisioning Resources with Scripts. 2. ed. New York: TechPress, 2020.

SOMMERVILLE, I. Engenharia de Software. 9. ed. São Paulo: Pearson, 2011.

VON HIPPEL, E. Democratizing Innovation. MIT Press, 2005.

WEBLOGIC. Weblogic Documentation. [S. l.], 1 jan. 2008. Disponível em: https://docs.oracle.com/middleware/1212/wls/. Acesso em: 10 out. 2022.

Documento Digitalizado Público

Artigo Final de TCC Nicolas G. Teixeira - Autologic: uma solução para o provisionamento automatizado do Weblogic

Assunto: Artigo Final de TCC Nicolas G. Teixeira - Autologic: uma solução para o provisionamento automatizado do Weblogic
Assinado por: Edgar Noda
Tipo do Documento: Comprovante
Situação: Finalizado
Nível de Acesso: Público
Tipo do Conferência: Documento Digital

Documento assinado eletronicamente por:

- **Edgar Noda**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 01/10/2024 09:09:41.

Este documento foi armazenado no SUAP em 01/10/2024. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1801860
Código de Autenticação: a96fed2d29