DOCUMENTO DE ARQUITETURA

deploy-utils

Sistema para automatização e rastreabilidade do processo de implantação de releases

Sumário

Introdução2

Requisitos [3](#_Toc329739055)

Mecanismos Arquiteturais4

Visão dos Componentes5

Estrutura de Arquivos7

Introdução

Este documento apresenta uma visão geral sobre a arquitetura da solução, abordando as decisões de projeto, tecnologias utilizadas, dependências, relação entre seus componentes, etc. Para melhor compreensão, recomenda-se a leitura prévia dos manuais de usuário e administrador, bem como o README no diretório raiz do projeto (contém o manual de instalação do servidor e dos agentes).

As seções subsequentes apresentarão os seguintes tópicos de forma detalhada:

* Requisitos
* MECANISMOS ARQUITETURAIS
* VISÃO DOS cOMPONENTES
* ESTRUTURA DE ARQUIVOS

Requisitos

O projeto foi originado da necessidade de desburocratizar o processo de implantação de software e minimizar os conflitos entre equipes de desenvolvimento e operações, favorecendo, simultaneamente, um ciclo de releases ágil e um maior controle sobre o ambiente computacional. O desenvolvimento da solução procurou empregar os conceitos da filosofia DevOps, atendendo aos seguintes objetivos específicos:

* AUTOMATIZAÇÃO

Possibilidade de implantação automática de código-fonte nos servidores de um ambiente computacional a partir de repositórios git.

* TRANSPARÊNCIA

Histórico detalhado de implantação de software no âmbito da organização.

* DEPLOY SIMPLIFICADO

Implantação simultânea de uma mesma versão de software em todos os hosts/instâncias do ambiente desejado. Suporte a diferentes tecnologias (asp, php, java, etc).

* DEBUG facilitado

Acesso a logs de diferentes servidores de aplicação por meio de uma interface centralizada.

* EXTENSIBILIDADE

Facilidade de criação de novos agentes de deploy e coleta de logs.

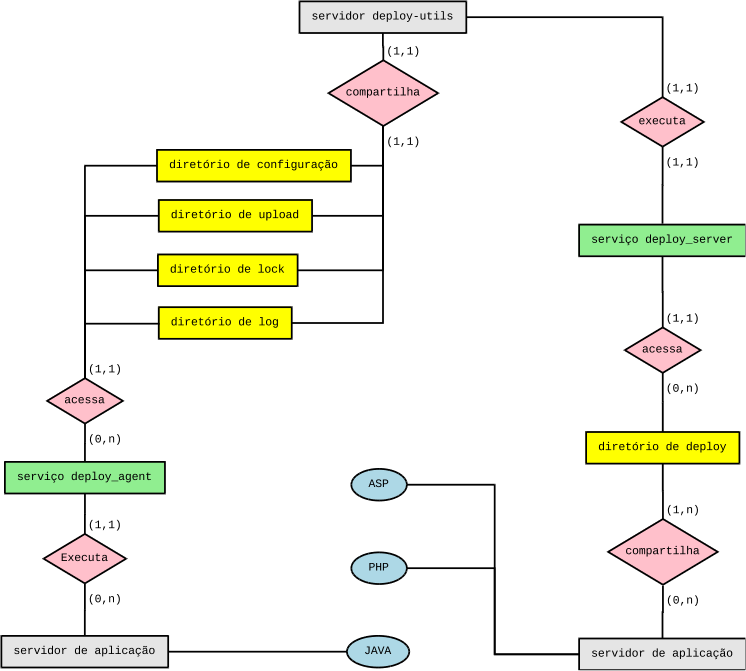
Mecanismos Arquiteturais

| Análise | Design | Implementação |
| --- | --- | --- |
| Apresentação | Interface de comunicação com o usuário do sistema | Servidor Web Apache HTTPD  Shell script: - Páginas CGI - /src/server/sh/functions.sh  (funções: web\_filter, parse\_multipart\_form, web\_header, web\_query\_history, web\_footer) |
| Controle | Utilização de daemons no servidor e agentes para execução de atividades periódicas ou solicitadas pelo usuário. | Shell script (arquivos .conf e .sh em geral) |
| Persistência | Script de consulta estruturada a arquivos texto com delimitadores e funções para abstração das operações básicas de criação, leitura, atualização e remoção de dados. | Shell Script: /src/common/sh/query\_file.sh (consulta estruturada a arquivos texto) /src/server/sh/functions.sh (funções:  add\_login - usuários delete\_login - usuários add\_group - grupos delete\_group - grupos membership - grupos members\_of - grupos subscribe - grupos unsubscribe - grupos chk\_permission - permissões add\_permission - permissões delete\_permission - permissões clearance - permissões editconf - configurações  ) /src/common/sh/functions.sh (função write\_history – histórico) |
| Banco de Dados | Arquivos texto com delimitadores foram utilizados para armazenar dados tabulares em lugar de um SGBD tradicional. | Histórico de deploy e tabela de permissões utilizam o padrão CSV; arquivos de usuários e grupos estão no padrão utilizado pelas diretivas AuthUserFile e AuthGroupFile do Apache; arquivos .conf são shell scripts na forma “chave=valor”. |

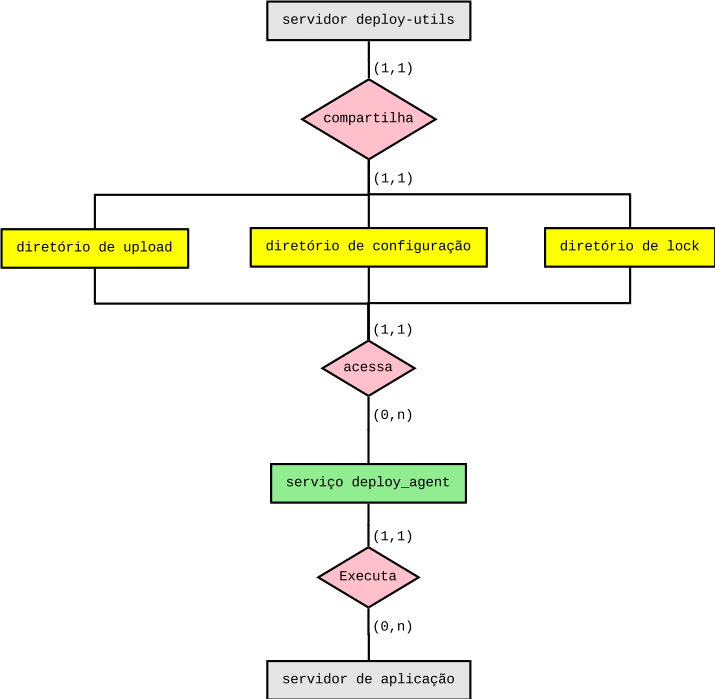
Visão dos Componentes

A comunicação entre o servidor deploy-utils, os agentes remotos e os servidores de aplicação ocorre através de diretórios compartilhados por meio dos protocolos CIFS ou NFS, conforme ilustram os diagramas de entidade-relacionamento abaixo para os processos de deploy de pacotes/páginas e coleta de logs, respectivamente:

**Deploy de pacotes/páginas**

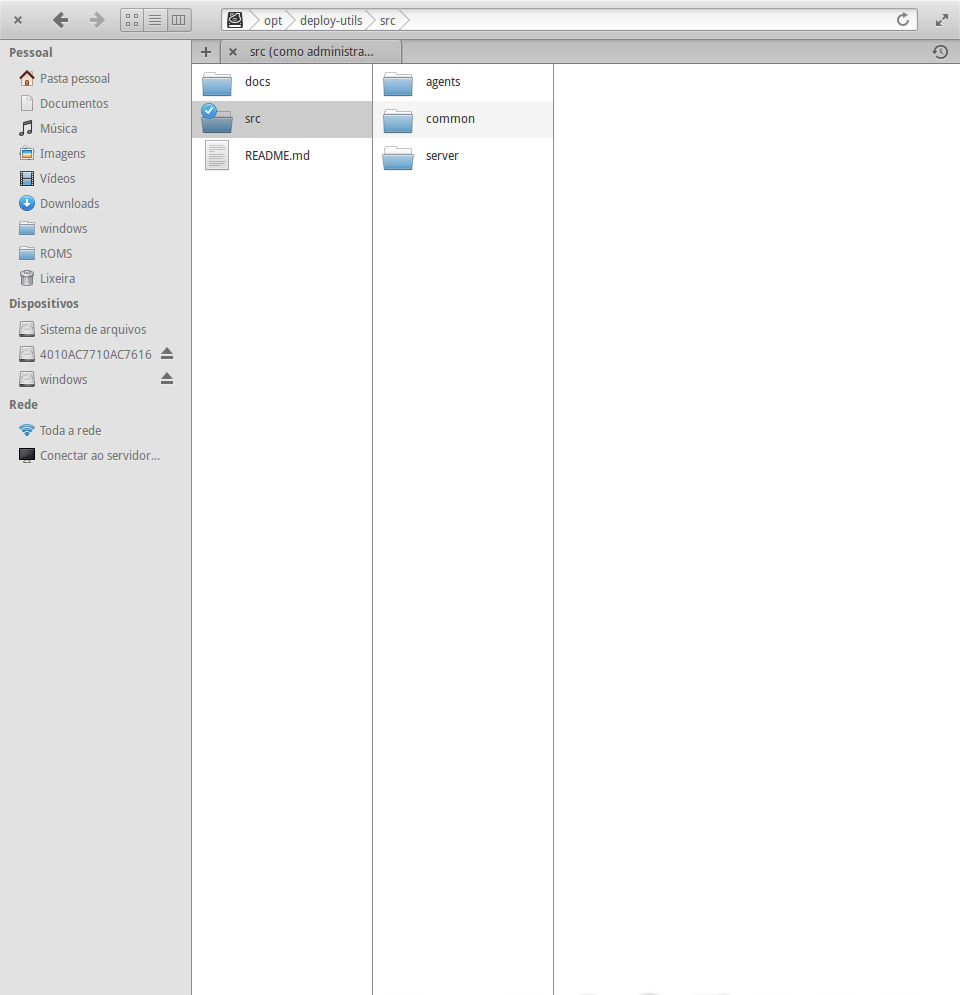


**Coleta de Logs**



Estrutura de Arquivos

O código-fonte foi organizado em três grandes grupos, sob os diretórios “server”, “agents” e “common”, como mostra a figura abaixo:

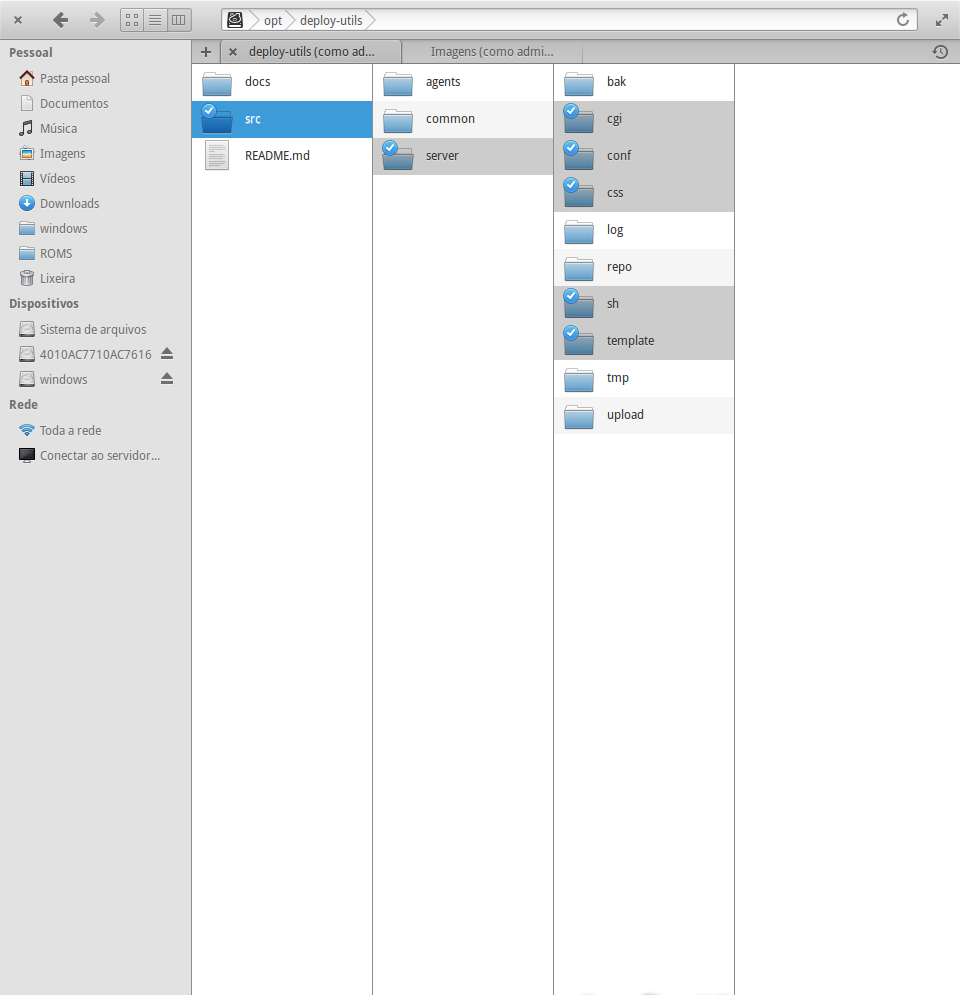


As subseções a seguir detalham a estrutura de arquivos sob cada grupo, considerando os caminhos utilizados por uma instalação default:

* Servidor
* AGENTES
* ARQUIVOS COMUNS

**Servidor**

Sob o diretório “server” encontram-se os scripts, configurações e templates utilizados majoritariamente pelo servidor deploy-utils, a saber:



* **cgi:** Scripts .cgi que compõem a interface de usuário. Os arquivos são executados pelo servidor web apache.
  + À exceção do script ‘table\_data.cgi’, cada arquivo representa uma tela do sistema e corresponde a um link no rodapé da interface web.
  + O script ‘table\_data.cgi’ é invocado pelas funcionalidades de relatório, tais como “Histórico de Deploy” e “Busca Avançada”. Sua função é padronizar a exibição dos resultados da consulta ao histórico de deploy global.
* **conf:** Contém arquivos de configuração relacionados às diversas tarefas desempenhadas pelo servidor deploy-utils.
  + O arquivo global.conf armazena as variáveis mais importantes do servidor, motivo pelo qual sua edição direta não é recomendada. Alternativamente, alguns valores default poderão ser sobrescritos através do arquivo user.conf
  + Os arquivos ‘passwords’, ‘groups’ e ‘permissions’ são utilizados pela interface web para autorizar ou conceder acesso aos recursos do sistema. O arquivo permissions é um csv na forma abaixo:

subject\_type;subject\_name;resource\_type;resource\_name;permission;

* + O subdiretório ‘apps’ contém os parâmetros de deploy para as aplicações interpretadas (como ASP e PHP).
  + O subdiretório ‘agents’ armazena os parâmetros de execução de agentes para os hosts que executam o serviço ‘deploy\_agent’.
* **css:** Contém a folha de estilos aplicada às páginas da interface web.
* **sh:** Contém cabeçalhos e funções a serem carregados por todos os scripts sob o diretório ‘server’, bem como executáveis internos do servidor:
  + O arquivo functions.sh contém as funções comuns do servidor.
  + O arquivo include.conf carrega as funções comuns e valida configurações importantes. É carregado no início de todos os executáveis do servidor.
  + O executável setup.sh realiza a configuração ou instalação/reparação do sistema, adicionando o virtualhost do apache e o serviço deploy\_server, atribuindo permissões, redefinindo parâmetros, dentre outros.
  + O executável server\_tasks.sh é o daemon invocado pelo serviço deploy\_server.
  + O executável deploy\_auto.sh é responsável pelo processamento assíncrono da fila de deploy (deploy\_queue). É invocado pelo daemon diretamente.
  + O executável deploy\_pages.sh é o comando responsável por simular ou efetuar os deploys de código-fonte das aplicações interpretadas (ex: ASP, PHP) a partir de revisões (commits ou tags) disponibilizadas nos respectivos repositórios git. É invocado pelo script deploy\_auto.sh, mas também pode ser executado diretamente por um administrador local caso necessário.
* **template:** armazena modelos de arquivos diversos:
  + Os arquivos htaccess, vhost e vhost.ssl são modelos de configuração do servidor web apache utilizados pelo script setup.sh.
  + O arquivo service.template é um modelo de script de inicialização do serviço deploy\_server. É utilizado pelo executável setup.sh para a instalação do arquivo /etc/init.d/deploy\_server.
  + Os demais arquivos são templates de configuração do próprio sistema deploy-utils. São utilizados para testar a integridade dos arquivos de configuração correspondentes, auxiliando na prevenção de vulnerabilidades como execução de comandos arbitrários ou carregamento de variáveis inválidas.

Outros diretórios são criados no servidor deploy\_utils durante a instalação, ou execução do sistema. Abaixo uma descrição detalhada, bem como os caminhos default:

* **bak (/opt/deploy-utils/src/server/bak):** armazena backups criados pelo comando deploy\_pages.sh antes da implantação de cada nova release. O último backup da aplicação em um ambiente é retido para fins de rollback automático nos hosts necessários em caso de interrupção do processo de deploy ou para downgrade após uma implantação bem-sucedida, conforme a conveniência do usuário.
* **lock (/var/lock/deploy-utils):** armazena lockfiles temporários que previnem a execução simultânea de tarefas sensíveis por processos concorrentes. Ex: checkout de repositórios git, escrita no histórico de deploy global, inicialização de serviços, etc.
* **log (/opt/deploy-utils/src/server/log):** armazena o histórico de deploy global e os logs de deploy e do serviço deploy\_server:
  + O arquivo service.log contém informações sobre o processamento da fila de deploy.
  + Os logs de deploy ficam hierarquizados sob o subdiretório ‘sistemas’. Os níveis subsequentes são ‘ambiente’, ‘aplicação’ e ‘deploy\_id’.
  + O histórico de deploy global é um csv na forna abaixo:

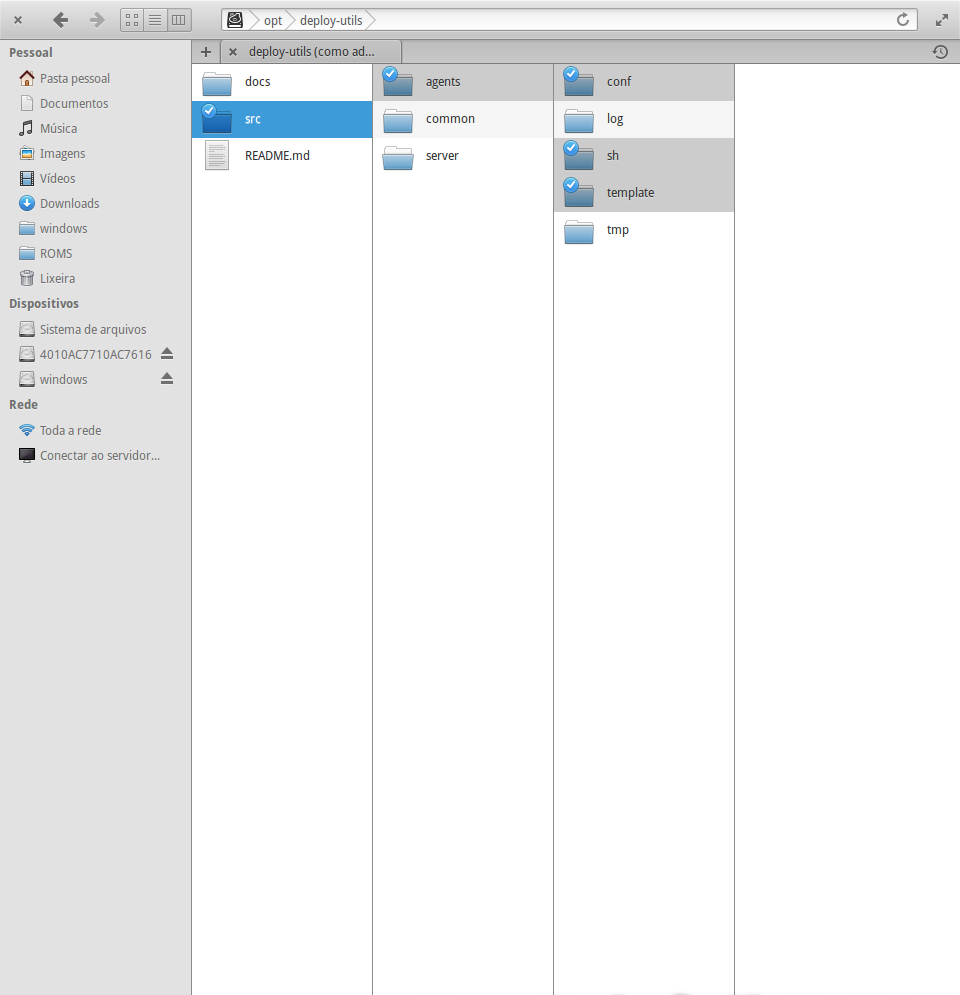
Dia;Mes;Ano;Hora;Usuario;Sistema;Revisao;Ambiente;Host;Observacao;Flag;

* **repo (/opt/deploy-utils/src/server/repo):** armazena os repositórios git a partir dos quais o comando deploy\_pages.sh realiza o checkout de novas releases de aplicações para deploy.
* **tmp (/opt/deploy-utils/src/server/tmp):** armazena a fila de deploys e os diretórios temporários utilizados por diversos executáveis do servidor.
  + Cada subdiretório criado tem como nome o PID do processo correspondente, a fim de garantir isolamento entre os arquivos temporários de cada processo.
  + O diretório tmp também é o local padrão para a fila de deploys (deploy\_queue). Trata-se de um named pipe responsável por receber as solicitações de deploy originadas na interface web através da tela deploy\_pages.cgi. Cada linha do arquivo é consumida pelo script deploy\_auto.sh, que, por sua vez, invoca o comando deploy\_pages.sh com os argumentos de deploy correspondentes. A saída do comando é redirecionada para um arquivo temporário do processo originado pelo servidor web, permitindo o acompanhamento da tarefa pelo usuário final. Esta abordagem foi escolhida por questões de segurança, uma vez que os deploys podem ser solicitados via web sem que o usuário “apache” (ou “www-data” no Debian) precise ser adicionado ao arquivo /etc/sudoers.
* **upload (/opt/deploy-utils/src/server/upload):** armazena uma árvore de diretórios cujos níveis são “ambiente”, “servidor\_aplicacao”, “host” e “aplicacao”. O último nível pode conter os subdiretórios “deploy” e/ou “log”.
  + Os diretórios “deploy” são populados quando um usuário efetua o upload de um pacote de determinada aplicação por meio da tela “deploy\_packages.cgi”. Em seguida, um ou mais agentes de deploy remotos correspondentes àquele ambiente realizam o deploy da aplicação em seus respectivos hosts.
  + Os diretórios “log” funcionam de forma distinta: seu conteúdo é atualizado regularmente pelos respectivos agentes de coleta de log, possibilitando o download de arquivos do servidor de aplicação selecionado pelo usuário através da tela “app\_logs.cgi”.

Além dos diretórios detalhados acima, há também os pontos de montagem temporários utilizados pelo processo de deploy de código-fonte, sob o caminho "/mnt/deploy\_${app}\_${host}". Esses diretórios são criados e removidos automaticamente pelo comando deploy\_agent.sh quando precisa realizar operações de leitura e escrita em diretórios compartilhados pelo servidor de aplicação.

**Agentes**

Sob o diretório “agents” encontram-se os scripts, configurações e templates utilizados exclusivamente pelo serviço deploy\_agent, instalado nos hosts de servidores de aplicação.



* **conf:** Contém parâmetros de configuração global do serviço deploy\_agent. O arquivo global.conf armazena as variáveis sensíveis do agente, motivo pelo qual sua edição direta não é recomendada.
* **sh:** Contém os agentes propriamente ditos (ex: jboss\_4\_5.sh e wildfly\_8.sh) e outros executáveis utilizados pelo serviço deploy\_agent:
  + O executável setup.sh realiza a configuração ou instalação/reparação do agente, adicionando o serviço deploy\_agent, atribuindo permissões, redefinindo parâmetros, dentre outros.
  + O executável agent\_tasks.sh é o daemon invocado pelo serviço deploy\_agent.
  + O executável run\_agent.sh é invocado regularmente pelo daemon. Sua função é obter do servidor deploy-utils as configurações definidas através da tela “agent\_params.cgi”, verificando em seguida os subdiretórios de upload correspondentes à cada configuração. Caso as condições para a realização de uma tarefa sejam satisfeitas, as variáveis e funções pertinentes são exportadas e o agente correspondente é executado na sequência
  + Os executáveis jboss\_4\_5.sh e wildfly\_8.sh são agentes de deploy e coleta de logs para os servidores JBOSS 4/5 e WILDFLY 8, respectivamente. Novos agentes podem ser criados a partir do template disponível em /opt/deploy-utils/docs/agent\_template.txt.
* **template:** armazena modelos de arquivos diversos:
  + O arquivo service.template é um modelo de script de inicialização do serviço deploy\_agent É utilizado pelo executável setup.sh para a instalação do arquivo /etc/init.d/deploy\_agent.
  + O arquivo global.template é um modelo de configuração para o arquivo global.conf
  + O arquivo agent.template é um modelo de configuração genérico para a a construção (e validação) de templates de configuração de agentes.
  + Os demais arquivos são templates de configuração dos agentes atualmente disponíveis (jboss\_4\_5 e wildfly\_8).

Outros diretórios são criados durante a instalação do serviço deploy\_agent ou execução de agentes. Abaixo uma descrição detalhada, bem como os caminhos default:

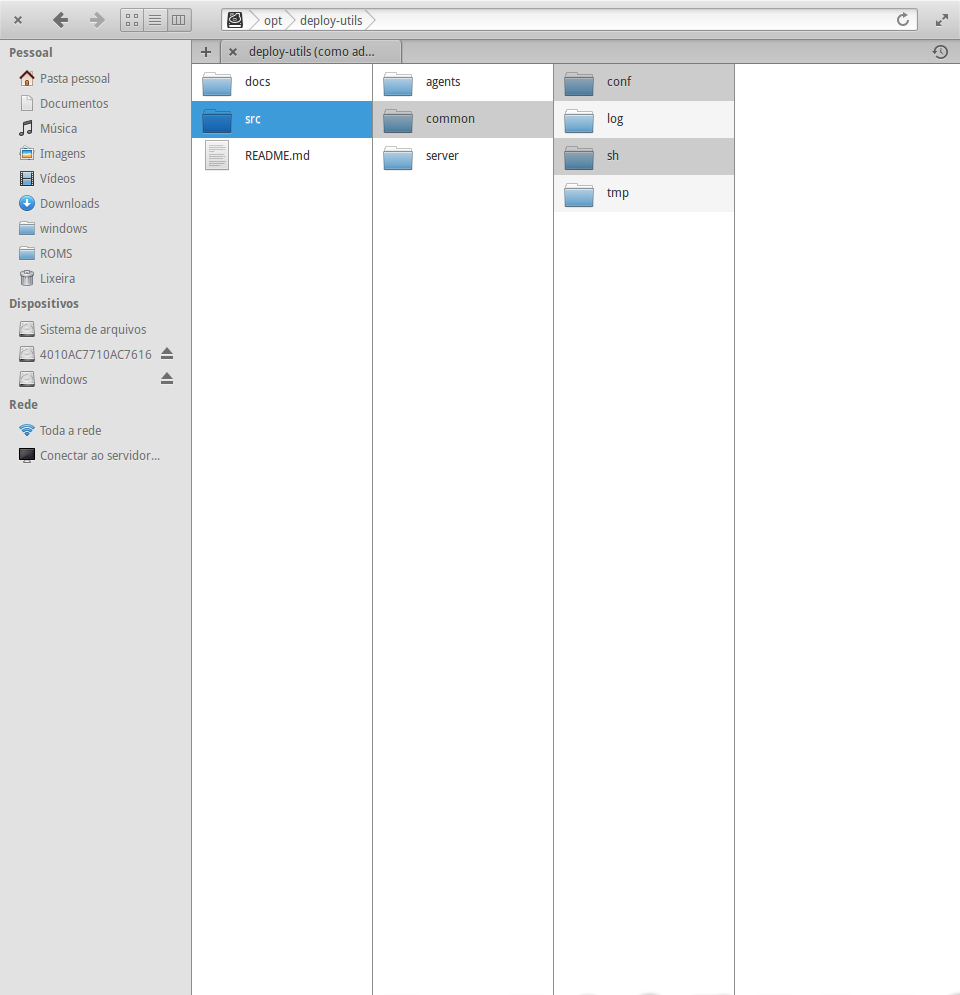
* **lock (/var/lock/deploy-utils):** armazena lockfiles temporários que previnem a execução simultânea de tarefas sensíveis por processos concorrentes. Ex: inicialização de serviços, agentes, etc.
* **log (/opt/deploy-utils/src/agents/log):** armazena o log do serviço deploy\_agent (service.log), que contém informações sobre a execução dos agentes de deploy e coleta de log habilitados para o servidor de aplicação.
* **tmp (/opt/deploy-utils/src/agents/tmp):** armazena os diretórios temporários criados durante a execução do serviço deploy\_agent. Cada subdiretório criado tem como nome o PID do processo correspondente, a fim de garantir isolamento entre os arquivos temporários de cada processo. As exceções são os agentes, que utilizam o diretório temporário do processo pai, pertencente ao comando run\_agent.sh.

Além dos diretórios locais acima, o serviço deploy\_agent precisa acessar ou escrever dados em diretórios compartilhados pelo servidor deploy-utils. Isso é feito por meio de pontos de montagem no diretório /mnt. São eles:

* **deploy\_conf (/mnt/deploy\_conf):** corresponde ao diretório “/opt/deploy-utils/src/server/conf/agents” do servidor deploy-utils. Contém subdiretórios correspondentes a cada host onde há um serviço deploy\_agent em execução. É utilizado para acessar parâmetros de configuração de agentes criados pela tela “agent\_params.cgi” da interface web.
* **deploy\_lock (/mnt/deploy\_lock):** corresponde ao diretório “/var/lock/deploy-utils” do servidor deploy-utils. É utilizado principalmente pelo agente de deploy de pacotes para criar a trava de edição do histórico de deploy global. Adicionalmente, o comando run\_agent.sh cria nesse diretório lockfiles com o padrão de nomenclatura “run\_agent\_${host}\_${pid}”, a fim de que o servidor deploy-utils impeça a deleção de configurações ou diretórios de aplicação durante seu acesso por um agente.
* **deploy\_log (/mnt/deploy\_log):** corresponde ao diretório “/opt/deploy-utils/src/server/log” do servidor deploy-utils. É acessado por agentes durante a execução de deploy de pacotes para a escrita dos logs de deploy nos subdiretórios correspondentes (sob o caminho “sistemas”) e para criar entradas no histórico de deploy global.
* **deploy\_upload (/mnt/deploy\_upload):** corresponde ao diretório “/opt/deploy-utils/src/server/upload” do servidor deploy-utils. É acessado por agentes para verificação de novos pacotes nos diretórios “deploy” e para atualização do conteúdo dos diretórios “log”, conforme já explicado na seção correspondente ao servidor deploy-utils.

**Arquivos comuns**

Sob o diretório “common” encontram-se os scripts e configurações comuns a todo o sistema:



* **conf:** armazena o arquivo include.conf, que contém configurações comuns a todo o sistema. Sua edição direta não é recomendável, uma vez que pode produzir efeitos colaterais em todo o sistema. No entanto, algumas customizações são possíveis **antes** da instalação do servidor deploy-utils e seus respectivos agentes na rede da organização (nesse caso, o arquivo customizado deverá ser distribuído para todos os hosts necessários), conforme segue:
  + *web\_context\_path:* é o contexto a partir do qual a interface web do sistema será acessada. **Sua alteração posterior à instalação quebrará os links para os logs de deploy já gravados no histórico de deploy global**.
  + *regex\_ambiente:* é a expressão regular que define os ambientes de deploy possíveis, na forma abaixo:

ambiente1|ambiente2|ambiente3|ambiente4

**Sua edição incorreta pode provocar problemas na realização de deploys, visualização de logs e do histórico**. O template “/opt/deploy-utils/src/server/template/app.template” e os arquivos de configuração sob o caminho “/opt/deploy-utils/src/server/conf” também precisarão ser modificados de forma correspondente.

* + *regex\_hosts\_${ambiente}:* este conjunto de expressões regulares poderá ser modificado para assegurar que os deploys sejam realizados nos hosts adequados, obedecendo a um padrão de nomenclatura de máquinas para cada ambiente. Caso não haja qualquer padrão de nomes, o valor abaixo poderá ser utilizado com segurança:

“($regex\_host[ ,]?)+”

**Sua edição incorreta pode provocar problemas na realização de deploys.**

* **sh:** armazena scripts comuns a todo o sistema.
  + O arquivo functions.sh contém as funções comuns do sistema, a fim de padronizar diversas operações importantes, tais como a formatação de mensagens de log, criação e remoção de lockfiles, validação de arquivos e variáveis, adição de entradas no arquivo de histórico e criação de diretórios de log de deploy.
  + O arquivo include.sh é um cabeçalho utilizado pelos executáveis do sistema. É responsável carregar os arquivos include.conf e functions.sh, além de atribuir as variáveis “src\_dir”, “install\_dir” e “PATH”.
  + O arquivo query\_file.sh é um script executável de consulta estruturada a arquivos de texto. Seus argumentos se assemelham à sintaxe da linguagem SQL. Foi criado para facilitar o acesso a dados do histórico de deploy e da tabela de permissões, dentre outros.
  + O arquivo reconfigure.sh é um script utilizado redefinir os parâmetros de configuração default do sistema, fazendo um backup das configurações anteriores. Pode ser executado diretamente, ou a partir dos scripts de setup dos diretórios “server” e “agents”, com a opção “--reconfigure”.

Além dos diretórios “conf” e “sh”, a instalação do sistema também cria os diretórios “log” e “tmp” por padrão. Atualmente, seu uso é restrito ao script query\_file.sh.