Agendamento de rotinas com CronTab ou Airflow/Schedule

## 

## CronTab

### O que é o Crontab?

O crontab é um utilitário do sistema operacional Unix que permite agendar tarefas para serem executadas automaticamente em intervalos regulares. É um recurso muito útil para tarefas como backups automatizados, atualizações de software e outras tarefas rotineiras.

Cada usuário do sistema pode ter seu próprio arquivo crontab, que contém uma lista de tarefas agendadas. Cada tarefa é descrita por uma linha no arquivo crontab que especifica quando a tarefa deve ser executada e qual comando deve ser executado.

A sintaxe básica para uma entrada no crontab é composta por seis campos:

\* \* \* \* \* comando

Os seis campos representam, na ordem:

1. Minuto (0-59);
2. Hora (0-23);
3. Dia do mês (1-31);
4. Mês (1-12);
5. Dia da semana (0-7, onde ambos 0 e 7 representam domingo); e
6. Comando a ser executado.

Os asteriscos nas primeiras cinco posições representam "todos", então, por exemplo, uma entrada \* \* \* \* \* significa que o comando será executado a cada minuto.

Você também pode especificar intervalos específicos, por exemplo, para executar uma tarefa todos os dias às 9h, você poderia usar a seguinte entrada no arquivo crontab:

0 9 \* \* \* comando

Você pode usar o comando crontab -e para editar seu arquivo crontab e adicionar ou remover tarefas agendadas. O crontab verifica periodicamente seu arquivo e executa qualquer tarefa agendada de acordo com as especificações.

### Executando arquivos Python com o Crontab

Aqui está um guia detalhado para criar um script Python e agendá-lo com o cron do Linux:

1. Escreva o script Python: Primeiro, escreva o script Python que você deseja executar periodicamente. Certifique-se de que ele possa ser executado sem problemas a partir do terminal. Salve-o com uma extensão .py.

Dê permissão de execução ao script: Abra o terminal e navegue até o diretório onde o script está localizado. Execute o seguinte comando:  
chmod +x nome\_do\_arquivo.py

Edite o arquivo crontab: No terminal, execute o seguinte comando para editar o arquivo crontab:  
crontab -e

Adicione uma tarefa ao crontab: Adicione a seguinte linha ao final do arquivo crontab:  
\* \* \* \* \* /usr/bin/python3 caminho/para/o/script.py

A primeira seção de cinco asteriscos representa a hora, minuto, dia do mês, mês e dia da semana, respectivamente, para a execução do script. Por exemplo, se você deseja executar o script todos os dias às 9h, a linha seria:  
0 9 \* \* \* /usr/bin/python3 caminho/para/o/script.py

1. Salve e saia: Salve o arquivo crontab e saia do editor. O cron agora irá executar o script periodicamente de acordo com a programação especificada.

Observação: o caminho para o interpretador Python pode ser diferente em sua máquina. Verifique o caminho correto usando o comando which python3 no terminal.

### Crie um ETL utilizando o Crontab

Vamos criar um passo a passo simples de como construir uma ETL (Extração, Transformação e Carregamento) em Python usando o crontab:

1. Escreva o script de ETL: Primeiro, escreva o script Python que extrai dados de uma fonte, realiza transformações nesses dados e os carrega em um destino. Por exemplo, você pode extrair dados de um banco de dados, realizar algumas transformações nele (como agregar ou filtrar dados) e carregá-lo em outro banco de dados ou em um arquivo CSV.
2. Salve o script: Salve o script em um arquivo com uma extensão .py.

Dê permissão de execução ao script: Abra o terminal e navegue até o diretório onde o script está localizado. Execute o seguinte comando:  
chmod +x nome\_do\_arquivo.py

Edite o arquivo crontab: No terminal, execute o seguinte comando para editar o arquivo crontab:  
crontab -e

Adicione uma tarefa ao crontab: Adicione a seguinte linha ao final do arquivo crontab:  
0 0 \* \* \* /usr/bin/python3 caminho/para/o/script.py

1. Neste exemplo, o script será executado todos os dias à meia-noite (0 horas, 0 minutos).
2. Salve e saia: Salve o arquivo crontab e saia do editor. O cron agora irá executar o script periodicamente de acordo com a programação especificada.

### Desvantagens do CronTab

Dentre as desvantagens do crontab, podemos citar:

1. Granularidade limitada: o crontab permite agendar tarefas com uma granularidade máxima de um minuto, o que pode não ser suficiente para algumas aplicações que precisam de tarefas mais frequentes;
2. Falta de recursos de gerenciamento de erros: o crontab não fornece recursos para gerenciar erros ou notificar falhas, então você precisa escrever seu próprio código para lidar com esses casos;
3. Falta de recursos de agendamento flexíveis: o crontab é limitado na sua capacidade de agendar tarefas de maneira flexível, com base em eventos específicos ou em outras condições;
4. Falta de recursos de gerenciamento de logs: o crontab não oferece recursos para coletar e gerenciar logs das tarefas agendadas, o que pode ser um problema para o monitoramento e depuração de problemas;
5. Limitações de escalabilidade: o crontab não é projetado para lidar com uma grande quantidade de tarefas agendadas, o que pode ser um problema em cenários de alta demanda;
6. Falta de recursos de gerenciamento centralizado: O crontab é um recurso de nível de usuário e não fornece recursos para gerenciar tarefas agendadas de maneira centralizada em um sistema de vários usuários; e

Em geral, o crontab é uma ferramenta útil para agendar tarefas simples, mas pode ser insuficiente para aplicações mais complexas ou exigentes. Nesses casos, é recomendável considerar alternativas, como o uso de ferramentas de agendamento de tarefas mais avançadas, como o Apache Airflow.

## Apache Airflow

### O que é o Apache Airflow?

O Apache Airflow é uma plataforma de fluxo de trabalho de código aberto que permite a criação, agendamento e monitoramento de pipelines de trabalho. Ele permite que você defina tarefas (conhecidas como "operações" ou "atividades") e especifique as dependências entre elas para formar um fluxo de trabalho. Além disso, o Airflow fornece recursos avançados de agendamento, como agendamento baseado em eventos e gerenciamento de falhas, tornando-o uma alternativa poderosa ao crontab.

Aqui estão algumas das vantagens do uso do Apache Airflow:

1. Agendamento avançado: o Airflow permite agendar tarefas de maneira mais flexível do que o crontab, incluindo agendamento baseado em eventos e agendamento dinâmico;
2. Monitoramento e gerenciamento de erros: O Airflow fornece recursos para monitorar e gerenciar erros nas tarefas agendadas, incluindo notificações por e-mail e registro de logs;
3. Integração com outras ferramentas: O Airflow fornece integrações com outras ferramentas e serviços, como o Apache Hive, o Apache Spark e o Amazon S3, tornando-o uma plataforma completa para ETL e outros processos de trabalho; e
4. Centralização e escalabilidade: O Airflow permite gerenciar tarefas agendadas de maneira centralizada, tornando-o uma solução escalável e de alto desempenho para aplicações de fluxo de trabalho.

Aqui estão algumas das desvantagens do uso do Apache Airflow:

1. Complexidade: O Airflow é uma ferramenta mais complexa do que o crontab e pode requerer um período de aprendizado e configuração mais longo;
2. Requisitos de hardware: O Airflow pode ser mais exigente em termos de recursos de hardware do que o crontab, devido à sua natureza escalável e centralizada; e
3. Dependência de outras tecnologias: O Airflow depende de outras tecnologias, como o Apache Celery e o Apache SQLAlchemy, para funcionar, o que pode aumentar a complexidade da instalação e configuração.

Para começar a usar o Apache Airflow, você precisará instalar o software em seu sistema e configurar o ambiente. Isso inclui a instalação de dependências, como o Apache Celery e o Apache SQLAlchemy, bem como a configuração do banco de dados que armazenará as informações do fluxo de trabalho.

Uma vez que o ambiente está configurado, você pode criar tarefas (ou "operações") no Apache Airflow usando Python. Cada tarefa é definida como uma classe Python que herda de uma classe base fornecida pelo Apache Airflow. Cada tarefa define as ações que devem ser executadas, bem como as dependências entre as tarefas.

Finalmente, você pode agendar as tarefas no Apache Airflow usando o painel de administração web. Aqui, você pode definir horários para as tarefas, bem como especificar outras configurações, como notificações por e-mail em caso de falhas.

Em resumo, o Apache Airflow é uma plataforma poderosa para agendar tarefas, especialmente para aplicações que requerem agendamento avançado e integração com outras ferramentas. No entanto, requer mais conhecimento e tempo de configuração do que o crontab e pode ser mais exigente em termos de recursos de hardware.

# Referências

* Documentação oficial do Linux: <https://manpages.ubuntu.com/manpages/bionic/man5/crontab.5.html>
* Guia do usuário para crontab: <https://medium.com/totvsdevelopers/entendendo-o-crontab-607bc9f00ed3>
* Documentação oficial do Apache Airflow: <https://airflow.apache.org/docs/stable/>
* Guia do usuário para Apache Airflow: <https://gtoonstra.github.io/etl-with-airflow/>
* Artigo introdutório ao Apache Airflow: <https://towardsdatascience.com/a-complete-introduction-to-apache-airflow-b7e238a33df>