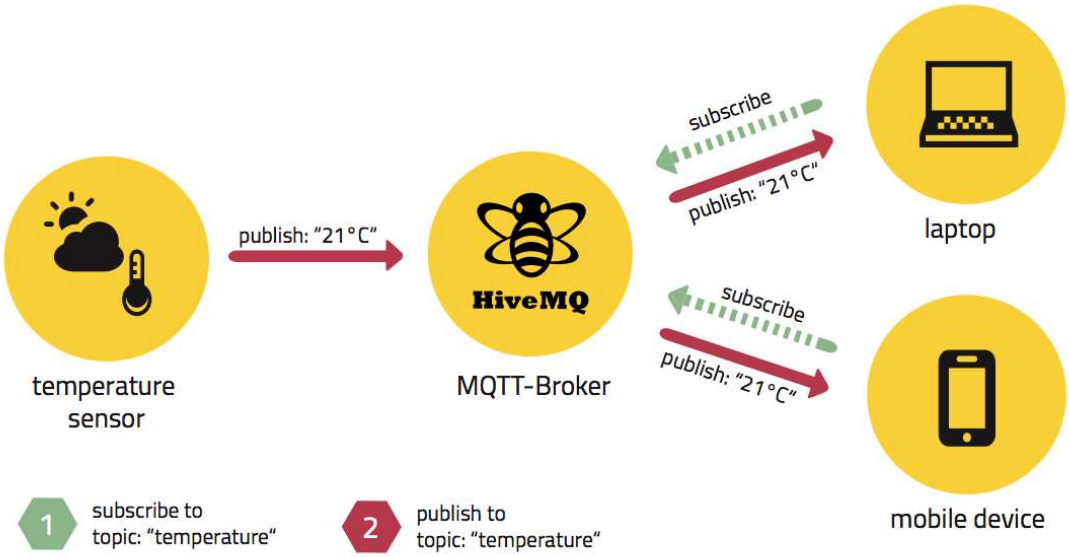

mosquitto


publish: "21°C"
HiveMQ
subscribe
publish: "21°C"
laptop
subscribe
publish: "21°C"
temperature sensor
MQTT-Broker
mobile device
1
subscribe to topic: "temperature"
2
publish to topic: "temperature"

Mosquitto MQTT Broker

Mosquitto é um agente de mensagens de código aberto leve que implementa MQTT versões 3.1.0, 3.1.1 e versão 5.0 Ele foi escrito em C por Roger Light e está disponível para download gratuito para Windows e Linux e é um projeto Eclipse.

MQTT Version 5.0 Support Notes:

A partir da versão 1.6, o corretor mosquitto suporta MQTT v5 além do MQTT v3.11. Você pode continuar a usar os clientes da versão 3.11 mais antigos com o corretor mais recente. O Mosquitto v2 introduziu algumas mudanças importantes que afetam particularmente os usuários iniciantes. Por padrão, ele requer autenticação e não escuta em um endereço de rede.

O seguinte arquivo de configuração simples fará com que o mosquit comece como nas versões anteriores:

listener 1883
allow_anonymous true

Instalando o Broker

Para usá-lo, você primeiro precisa instalá-lo. Nota especial de dezembro de 12-2020: A última versão 2.x difere consideravelmente das versões 1.6 anteriores na segurança de instalação e, portanto, para iniciantes, eu recomendo que você fique com as versões 1.x mais antigas por enquanto.

Iniciando e parando o Broker

Dependendo da instalação, ele provavelmente será iniciado automaticamente na inicialização do sistema. Embora isso seja muito desejável em ambientes de produção, é menos desejável em ambientes de teste.

Minha abordagem preferida é interromper o serviço mosquitto e iniciá-lo manualmente no prompt de comando. Isso lhe dá acesso ao console, que é inestimável para teste.

No Windows, você pode interromper o serviço se ele estiver em execução usando o painel de controle> admin> serviços. Você também pode usar o comando net:

net stop mosquito

No Linux o comando é:

sudo service mosquitto stop
sudo systemctl stop mosquitto.service

Começar a partir da linha de comando é a melhor opção durante o teste e, para isso, use:

mosquitto -v #start in verbose mode (Windows e Linux)

Para ver outras opções de início, use:
mosquitto -h

Por padrão, o broker começará a escutar na porta 1883.
Como alternativa, você pode usar uma opção de linha de comando para especificar a porta, por exemplo:
mosquitto -p 1884

Programas de clientes Mosquitto

O mosquitto install inclui os programas de teste do cliente. Existe um cliente assinante simples:**mosquitto_sub** e um cliente editor: **mosquitto_pub** . Eles são úteis para alguns testes rápidos.

Perguntas e respostas comuns

P- As mensagens são armazenadas no corretor? R- Sim, mas apenas temporariamente. Depois de enviados a todos os assinantes, eles são descartados.

P - Existe um limite para o tamanho da mensagem permitido por um corretor? A- MQTT impõe um limite máximo de tamanho de mensagem de 268.435.456 bytes. Você pode restringir o tamanho máximo da mensagem que o corretor aceitará usando: limite de tamanho_de_mensagem por exemplo message_size_limit 1000 no arquivo de configuração do mosquitto. As mensagens recebidas acima do limite são descartadas pelo corretor.

P- O corretor precisa de um arquivo de configuração para iniciar? A- Não

P- Qual é a configuração de persistência no arquivo de configuração do mosquitto? A- Quando habilitado, o corretor armazena dados temporários como conexões persistentes, mensagens retidas e últimas mensagens em um arquivo. Se o servidor for reiniciado, os valores serão restaurados.

Protocolo MQTT e estrutura

O **MQTT-** ***Message Queue Telemetry Transport*** (Transporte de Filas de Mensagem de Telemetria) é um Protocolo de Comunicação baseado na pilha TCP/IP, sendo extremamente útil para desenvolvimento de projetos de comunicação entre Máquina-Máquina (M2M), seu conceito de transmissão é do tipo Publicação/Assinatura, que será explicado mais adiante.

Desenvolvido por Andy Stanford-Clark da IBM e Arlen Nipper em 1999, sua aplicação de origem consistia em monitorar sensores em oleodutos de petróleo através de satélites, o MQTT foi liberado ao público de forma gratuita no ano de 2010. Em 29 de outubro de 2014 o protocolo se tornou padrão OASIS- *Organization for the Advancement of Structured Information Standards* (Organização para o Avanço dos Padrões de Informação Estruturada). Atualmente o protocolo está em sua versão 5.0 (3 de abril de 2019), com melhorias no que tange o relatório de erros, as assinaturas compartilhadas, as

propriedades da mensagem, expiração de mensagens, expiração da sessão, alias do tópico, função atraso na entrega e descoberta de funções permitidas.

O MQTT apresenta inúmeras qualidades e vantagens com relação a outros protocolos como o HTTP, mostrando-se ideal para comunicação entre dispositivos remotos em razão de sua qualidade de serviço, maior nível de segurança, facilidade de implementação, baixa alocação de banda, bibliotecas compatíveis com diversas linguagens de programação, seu conceito de distribuição, níveis de serviço de acordo com relevância da mensagem, garantias de entrega, variações de envio de dados, sendo que um dado pode ser enviado para nenhum, um, ou vários clientes, entre outras qualidades que o tornam ideal para projetos IoT.

Forma de comunicação

A publicação e recebimento de dados são feitos através de um servidor nomeado de Broker. Um cliente no papel de Pulicador ou Publisher, que é a pessoa que transmite a mensagem, escreve um tópico de destino da mensagem e o seu Payload (conteúdo da mensagem), essa mensagem então é transmitida ao Broker que será responsável por gerir e encaminhá-la ao Subscrito ou Subscriber previamente inscrito no tópico. Da mesma forma, quando um cliente quer se tornar um Subscrito em um determinado tópico, ele encaminha uma mensagem de solicitação ao Broker, que fará essa interligação entre cliente e tópico.

Podemos simplificar em:

- **Publicador/ Publisher:** Quem envia dados para um tópico, emissor.
- **Subscrito/ Subscriber:** Pessoa que está inscrita no tópico e recebe os dados, receptor.
- **Broker:** Intermédio de comunicação entre Publicador e Subscrito, responsável por receber, enfileirar e enviar as mensagens.
- **Payload:** Conteúdo da mensagem enviada.
- **Cliente/Client:** Elemento capaz de interagir com o Broker, seja para enviar, receber ou os dois.
- **Mensagem:** Pacote de dados trocados entre Clientes e Broker.
- **Tópico:** Endereço para o qual os dados serão encaminhados.
- **Unsubscribe:** Deixar de assinar um tópico.

Vale mencionar que um Cliente pode agir tanto como Publicador quanto subscrito. Uma grande vantagem do MQTT é o uso do Broker, pois ele garante a segurança entre Publicador e Subscrito através da comunicação indireta entre ambos.

Qualidade da Mensagem (QoS)

Nesse protocolo de comunicação as mensagens são definidas por qualidade, de forma a determinar os níveis de importância e necessidade de recebimento. A mensagem pode ser definida em três níveis diferentes, sendo eles:

QoS 0 (*at most once*)-A mensagem deve ser recebida no máximo uma vez, podendo ser recebida uma vez ou nenhuma, não há confirmação de recebimento.
QoS 1 (*at least once*)-A mensagem deve ser recebida pelo menos uma vez, podendo uma mesma mensagem ser recebida uma ou mais vezes, há confirmação de entrega de uma mensagem.
QoS 2 (*exactly once*)-A mensagem deve ser recebida uma única vez, confirmação de recebimento de uma única mensagem.

Lembrando que não existe qualidade melhor ou pior, isso vai depender da finalidade da mensagem que será enviada.

As pré-configurações da mensagem são definidas pelo Publicador e pelo Subscrito, sendo que cada um realiza as suas próprias configurações, podendo existir as seguintes situações:

Publicador	Subscrito	Mensagem recebida
QoS 0	QoS 0	QoS 0
QoS 0	QoS 1	QoS 0
QoS 0	QoS 2	QoS 0
QoS 1	QoS 0	QoS 0
QoS 1	QoS 1	QoS 1
QoS 1	QoS 2	QoS 1
QoS 2	QoS 0	QoS 0
QoS 2	QoS 1	QoS 1
QoS 2	QoS 2	QoS 2

Tópicos

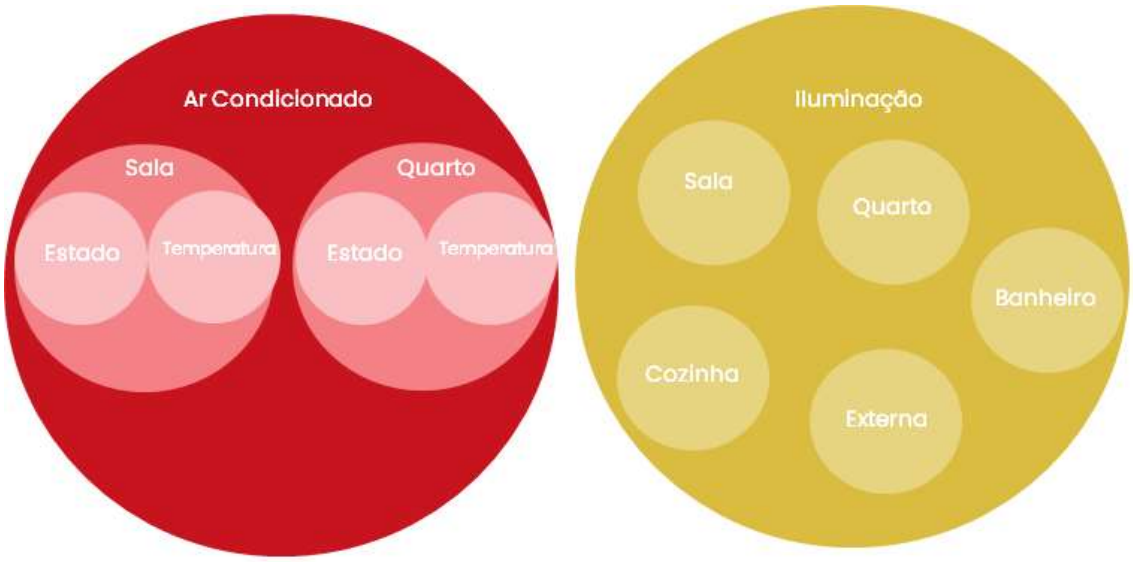
Um tópico é um endereço para qual uma mensagem será encaminhada, um Publicador cria o endereço e o Subscrito se inscreve naquele endereço para receber as informações, um Subscrito pode se inscrever em diversos tópicos simultâneos, enquanto um Publicador pode criar diversos tópicos e subtópicos de forma a organizar e categorizar as informações, assim como pastas no

computador. Um cliente também pode publicar em um tópico, desde que o mesmo não seja restrito. Abaixo eu fiz um exemplo de tópicos de uma automação residencial, de forma a elucidar melhor a explicação.

Objeto de análise	Tópico
Ar Condicionado	ArCondicionado/#
Estado do Ar Condicionado na Sala	ArCondicionado/Sala/Estado
Temperatura do Ar Condicionado na Sala	ArCondicionado/Sala/Temperatura
Estado do Ar Condicionado no Quarto	ArCondicionado/Quarto/Estado
Temperatura do Ar Condicionado no Quarto	ArCondicionado/Quarto/Temperatura
Iluminação	Iluminação/#
Iluminação na Sala	Iluminacao/Sala
Iluminação no Quarto	Iluminacao/Quarto
Iluminação na Cozinha	Iluminacao/Cozinha
Iluminação no Banheiro	Iluminacao/Banheiro
Iluminação Externa	Iluminacao/Externa

OBS: Devemos nos atentar ao uso de caracteres maiúsculos e minúsculos, existe essa diferenciação entre tópicos, um tópico ArCondicionado não é igual a um tópico arcondicionado.

Eu criei o tópico do Ar Condicionado e criei subtópicos dos cômodos com ar condicionado, em seguida criei outro tópico dentro do subtópico para definir a informação que quero receber, se é de temperatura ou do estado do mesmo. Assim como quando crio pastas dentro de uma pasta no computador, seria como se eu tivesse uma pasta chamada de Ar Condicionado, e dentro dessa pasta houvessem duas pastas, uma chamada de Sala e outra de Quarto, dentro da Sala haveriam mais duas pastas, sendo de Temperatura e Estado, na pasta quarto também haveriam duas pastas, igualmente de Temperatura e Estado.
O Diagrama de Venn dos tópicos que eu criei, pode ajudar na compreensão.



Além disso, ainda existem caracteres curingas para inscrição de tópicos, sendo o **+** e o **#**. Ambos permitem a inscrição em mais de um tópico de forma simplificada, sem precisar se inscrever em cada um individualmente, sendo o **+** utilizado no meio do endereço do tópico, e a **#** no final, como segue o exemplo com base na tabela acima:

Iluminacao/# → Se assino esse tópico, eu atomaticamente assino todos os tópicos de iluminação, sendo que receberei as informações de:
Iluminacao/Sala
Iluminacao/Quarto
Iluminacao/Cozinha
Iluminacao/Banheiro
Iluminacao/Externa

Guia rápido para o arquivo Mosquitto.conf com exemplos

Você pode configurar o corretor mosquitto usando um arquivo de configuração. O arquivo de configuração padrão é chamado **mosquitto.conf** e é usado pelo corretor mosquitto quando iniciado como um daemon Linux ou serviço do Windows.

Você encontrará o arquivo mosquitto.conf no diretório / etc / mosquitto no Linux e no diretório c: \ mosquitto \ no Windows. Nota: a instalação do Windows permite que você escolha o diretório.

Importante: para que as alterações no arquivo mosquitto.conf tenham efeito, você deve reiniciar o corretor mosquitto. No entanto, muitas alterações podem se tornar efetivas sem reiniciar o corretor e são marcadas com o comentário Reloaded on reload signal no manual.

MQTTv5 and MQTT v3.1.1

O corretor mosquitto suporta ambos os tipos de cliente e, portanto, algumas das configurações no arquivo de configuração afetarão apenas os clientes MQTTv5.

Ouvintes

É possível configurar um corretor mosquitto para escutar em várias portas ao mesmo tempo.

Isso é muito útil se você deseja que seu broker ofereça suporte a várias configurações de protocolo. Os mais comuns são:

MQTT

MQTT + SSL

MQTT + Websockets

MQTT + Websockets + SSL

A configuração padrão usa um ouvinte padrão que escuta na porta 1883.

Para configurar o broker para escutar em portas adicionais, você precisa criar escutas extras.

Se você olhar as configurações, verá que elas estão divididas em seções e algumas das configurações afetam toda a instância do broker, enquanto outras afetam um ouvinte específico.

Uma grande mudança introduzida na versão 1.5 permitiu que muitas definições de configuração que eram anteriormente globais fossem feitas por ouvinte.

No entanto, para manter a compatibilidade com versões anteriores, essa opção deve ser habilitada primeiro.

per_listener_settings [true | false]

Definir como verdadeiro afetará as seguintes opções de configuração.

password_file, acl_file, psk_file, allow_anonymous, allow_zero_length_clientid, auth_plugin, auth_opt_*, auto_id_prefix.

O mais importante é a capacidade de configurar opções de autenticação e controle de acesso por ouvinte e não globalmente, como acontecia antes da versão 1.5.

Default Settings

Todas as configurações têm uma configuração padrão que não está definida no arquivo de configuração, mas é interna ao mosquitto. As configurações no arquivo de configuração substituem essas configurações padrão.

Notas de configurações

Ao consultar o manual do Mosquitto.conf, você encontrará duas propriedades de configuração importantes. Eles são:

Global - Isso significa que se aplicam a todos os ouvintes

Recarregado no sinal de recarga. - As alterações podem ser implementadas enquanto o mosquitto está em execução, usando um reload.

Configurações globais

Essas configurações são globais e geralmente recarregadas no sinal de recarga.

Isso inclui registro e localização de arquivos. O registro é abordado com mais detalhes no tutorial de registro.

Eles também cobrem as configurações de persistência que permitem ao mosquitto manter mensagens e informações de status, como mensagens retidas, durante as reinicializações.

per_listener_settings [true | false]

allow_anonymous [true | false]

persistence [true | false]

persistence_file file name

persistence_location path

autosave_interval seconds

retain_available [true | false]

user username

A opção de usuário permite que você execute o Mosquitto como um usuário diferente (Linux) por padrão, ele é executado como o uso do Mosquitto. Configurações de restrição de mensagem Existem várias configurações de restrição de mensagens disponíveis.

Essas configurações são globais e afetam todos os ouvintes. Os mais importantes são:

max_packet_size value

message_size_limit limit -MQTT v5 clients

max_inflight_bytes count

max_inflight_messages count

max_queued_bytes count

max_queued_messages count

Configurações de autenticação

Permite que os usuários se conectem sem uma senha ou reforça a autenticação de nome de usuário / senha.

Isso pode ser configurado por ouvinte se per_listener_settings for verdadeiro. O padrão é verdadeiro, desde que nenhuma outra opção de segurança esteja presente. Se, por exemplo, um password_file ou psk_file for definido, o padrão é false.

allow_anonymous — [true | false]

Associadas à configuração de permissão anônima estão as configurações do arquivo de senha.

password_file file path

Se permitir anônimo for verdadeiro, você precisará criar um arquivo de senha e definir o caminho para o arquivo. Isso é abordado no tutorial de autenticação de nome de usuário / senha.

Isso pode ser definido por ouvinte e é recarregado no sinal de recarga.

Isso significa que você não precisa reiniciar o Mosquitto ao adicionar novos usuários ao arquivo de senha.

Suporte TLS / SSL

Fornecido por meio de certificados ou chaves pré-compartilhadas (PSK) e é configurável por ouvinte e não requer a configuração de per_listener_settings.

Isso é coberto com mais detalhes na configuração de SSL no Mosquitto.

Restrições de controle de acesso

Você pode configurar mosquitto para restringir o acesso a clientes usando ACL (listas de controle de acesso).

As restrições da lista de controle de acesso são definidas usando a configuração:

acl_file *file path*

e podem ser configurados por ouvinte se per_listener_settings for verdadeiro. Eles são recarregados no sinal de recarga.

Isso significa que as alterações feitas nas listas de controle de acesso podem ser aplicadas sem reiniciar o broker.

Consulte Usando e testando as restrições de ACL em mosquitto para obter mais detalhes

Por configurações de ouvinte

Existem muitas configurações que se aplicam aos ouvintes, independentemente da configuração if per_listener_settings.

Listener padrão

Esta escuta na porta 1883 por padrão e geralmente não requer configuração. No entanto, você pode definir o endereço que o ouvinte escuta usando a configuração bind_address e a interface usando a configuração bind_interface (somente Linux) e também o número da porta usando a configuração da porta.

Se você configurar o listener para usar websockets usando a configuração de protocolo, também poderá configurar o broker para agir como um servidor http simples e definir o diretório onde os arquivos estão localizados usando a configuração http_dir.

Observação: o manual não recomenda o uso do ouvinte padrão se você estiver configurando ouvintes adicionais.

A seguir está uma lista de outras configurações tiradas do manual que devem dar uma ideia do que pode ser configurado.

bind_address address

bind_interface device

http_dir directory

listener port [bind address/host]

max_connections count

maximum_qos count

max_topic_alias number -MQTTv5 only

mount_point topic prefix

port port number

socket_domain [ipv4 | ipv6]

protocol value (MQTT or websockets)

use_username_as_clientid [true | false]

websockets_log_level level

websockets_headers_size size

Configurações de ponte

O Mosquitto pode ser configurado para atuar como uma ponte, de modo que ele contará com mensagens para outro corretor - há uma seção inteira que cobre essas configurações.

As pontes também podem ser configuradas para usar autenticação e SSL.

Você pode encontrar mais detalhes na configuração do Mosquitto como um tutorial de ponte

Iniciando o Mosquitto - Notas

Ao iniciar o mosquitto para a linha de comando, a menos que você especifique um arquivo de configuração, nenhum é usado. Portanto

mosquitto

mosquitto -v

e outros comandos semelhantes iniciam o mosquitto sem usar um arquivo de configuração.

Se você instalar o mosquitto como um serviço no Windows, ele começará a usar o mosquitto.conf.

A instalação do Linux também configura o corretor Mosquitto para iniciar automaticamente usando o arquivo mosquitto.conf.

Importante: ao testar mosquitto, você precisa interromper a instância mosquitto que foi iniciada quando a máquina foi inicializada e, em seguida, iniciar sua própria instância a partir da linha de comando.

Editando o arquivo de configuração

Ao testar, eu recomendo que você crie um arquivo de configuração em sua pasta local e use-o.

Não recomendo que você copie o arquivo mosquitto.conf padrão, pois ele contém todas as configurações possíveis que estão comentadas e, se você fizer alterações, será difícil localizá-las.

No entanto, é muito útil para a documentação, pois contém ajuda para todas as configurações.

Você pode usar um editor de texto normal para editar o arquivo.

Se você usar, coloque todas as suas configurações no início do arquivo e use a seção comentada como documentação.

Se você editar as seções individuais, criei um script Python simples que exibirá apenas as configurações não comentadas do arquivo.

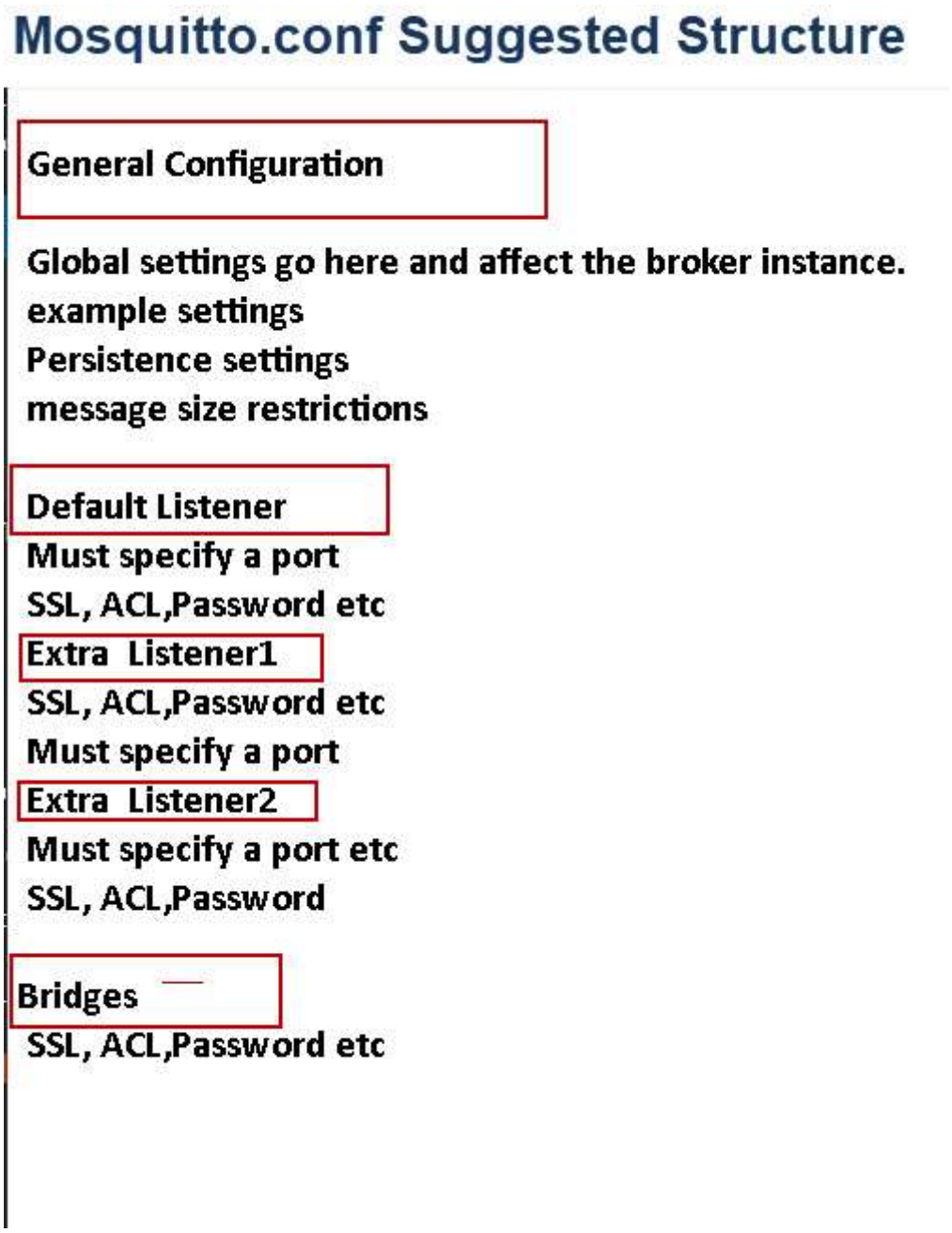
Ao executá-lo, você verá algo como a captura de tela abaixo

```
C:\Python34\steve\mqtt-tools>python checkconfig.py -f m1.conf
Using  C:\Python36\python.exe
args  3
Checking file  m1.conf
port 1883    #######Line Number 134
listener 1884    #######Line Number 299
protocol mqtt    #######Line Number 300
log_type error    #######Line Number 488
log_type warning    #######Line Number 489
log_type notice    #######Line Number 490
log_type information    #######Line Number 491
log_type all    #######Line Number 492
connection_messages true    #######Line Number 503
psk_file c:\mosquitto\certs\psk-file.txt    #######Line Number 558
acl_file c:\mosquitto\aclfile.example    #######Line Number 608
```

Estrutura do Arquivo

Não há estrutura imposta, embora certas configurações devam aparecer antes de outras, por exemplo, você deve especificar a configuração per_listener_settings antes de criar ouvintes.

A estrutura que utilizo é mostrada no esboço abaixo:



Exemplos de configurações

A melhor maneira de entender como usar o arquivo de configuração é ver alguns exemplos. O seguinte mostra algumas configurações típicas.

Nota: Eu não incluo registro ou persistência nestes exemplos para mantê-los curtos e simples, mas eu incluo meus arquivos reais.

Mosquitto Broker ouvindo em várias portas

O corretor mosquitto pode ser configurado para escutar em várias portas ao mesmo tempo.

No entanto, essa configuração não significa que você tem corretores virtuais. já que a maior parte da configuração é compartilhada.

Exemplo 1- Escuta nas portas 1883 e 1884

Ouvinte padrão da seção

port 1883

Ouvintes extras da seção

listener 1884

Exemplo 2- Escute nas portas 1883 e 1884 sem ouvinte padrão

Ouvinte padrão da seção

Ouvintes extras da seção

listener 1883

listener 1884

Exemplo 3- Escuta nas portas 1883 e 8883 (SSL)

Ouvinte padrão da seção

port 1883

Ouvintes extras da seção

listener 8883

Suporte SSL / TLS baseado em certificado

(Windows)

cafile c:\mosquitto\certs\ca.crt

keyfile c:\mosquitto\certs\server.key

certfile c:\mosquitto\certs\server.crt

(Linux)

cafile /etc/mosquitto/certs/ca.crt

keyfile /etc/mosquitto/certs/server.key

certfile /etc/mosquitto/certs/server.crt

Exemplo 4- Escuta nas portas 1883 e WebSockets (SSL)

Ouvinte padrão da seção

port 1883

Ouvintes extras da seção

listener 9001

protocol websockets

Exemplo 5 - Escute nas portas 1883 e 1884 com autenticação

Notas:

Global

per_listener_settings true

Ouvinte padrão da seção

port 1883

Ouvintes extras da seção

listener 1884

allow_anonymous false

password_file file path

Exemplo 6 - Restringir o número de conexões no ouvinte padrão para 1000

Global

max_connections 1000

Using MQTT Over WebSockets with Mosquitto

O que é Websockets e como funciona?

WebSocket é um protocolo de comunicação de computador, fornecendo canais de comunicação full-duplex em uma única conexão TCP / IP. Wiki

Ele está intimamente associado ao http, pois usa http para o estabelecimento da conexão inicial.

O cliente e o servidor se conectam usando http e, em seguida, negociam uma atualização de conexão para websockets, a conexão então muda de http para websockets.

O cliente e o servidor agora podem trocar dados binários full duplex pela conexão.

Por que usar MQTT em vez de Websockets?

MQTT sobre Websockets permite que você receba dados MQTT diretamente em um navegador da web.

Isso é importante porque o navegador da web pode se tornar a interface DE-facto para exibir dados MQTT.

O suporte de websocket MQTT para navegadores da web é fornecido pelo cliente JavaScript.

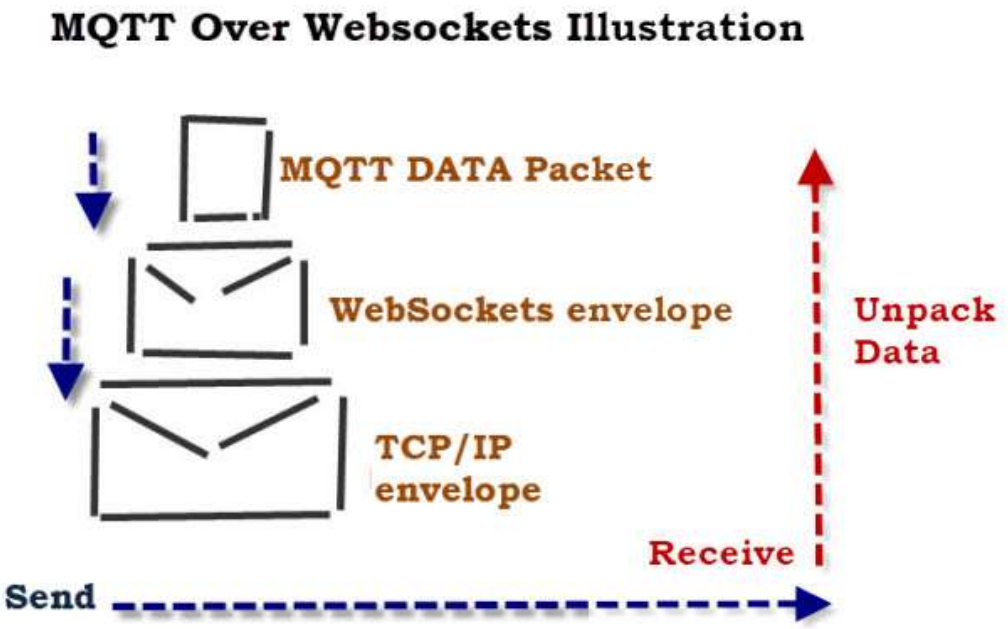
MQTT sobre Websockets vs MQTT.

No caso de MQTT sobre Websockets, a conexão de websockets forma um canal externo para o protocolo MQTT.

O broker MQTT coloca o pacote MQTT em um pacote websockets e o envia ao cliente.

O cliente descompacta o pacote MQTT do pacote websockets e o processa como um pacote MQTT normal.

Isso é ilustrado no diagrama abaixo:



Com o MQTT, o Pacote MQTT é colocado diretamente no Pacote TCP / IP.

Websockets e Mosquitto

Nota: Esta seção é deixada para referência. Os pacotes de instalação do Windows ainda não têm websockets. O pacote Linux instalado usando apt-get atualmente instala o mosquitto versão 1.4.12 e já possui suporte para websockets.

Os pacotes de instalação padrão do Mosquitto para Windows e Linux não incluem suporte para WebSockets.

Se você deseja testar WebSockets com MQTT, você precisa: Compile e instale seu próprio corretor Mosquitto com suporte para Websocket. Use um corretor on-line que tenha suporte para websocket. Existem instruções para compilar Mosquitto com websockets aqui (Linux) e aqui (windows). Notas de instrução para http://goochgooch.co.uk/2014/08/01/building-mosquitto-1-4/

As instruções do guia funcionam. mas eu precisava fazer algumas pequenas alterações, conforme detalhado aqui: Não foi possível encontrar libwebsockets-1.22-chrome26-firefox18.tar.gz, então usei em vez disso libwebsocckets-1.7.8.tar.gz. Peguei aqui https://libwebsockets.org/git/libwebsockets/

Não havia makefile, então tive que criar o diretório de compilação e executar o cmake. O arquivo ares.h estava faltando, então eu precisei instalá-lo de acordo com as instruções.Eu sempre me esquecia de executar comandos como sudo, o que significava que as cópias falhavam durante a execução do make.

Quando eu iniciei o corretor, ele não conseguiu encontrar libwebsockets.so.7, mas executando o comando

sudo ldconfig

Resolvi-o.

Websockets on Windows

Desde mosquitto 1.5.1, o suporte a websockets foi habilitado nos arquivos binários do Windows.

No entanto, quando você inicia o mosquitto, ele parece estar ouvindo na porta do websocket, mas não permite conexões.

mosquitto v 1.5.4 funciona com websockets. Aqui está um link para os downloads.

Configurando Websockets em seu próprio Mosquitto Broker

MQTT sobre Websockets geralmente usa a porta 9001, mas não é corrigido.

Você precisa fazer alterações no arquivo mosquitto.conf, adicionando o seguinte:

listener 9001

protocol websockets

Isso cria um ouvinte extra usando websockets e porta 9001.

Ao iniciar o corretor, você verá algo assim:

```
steve@steve-linux / $ mosquitto -c /etc/mosquitto/mosquitto.conf
1494527604: mosquitto version 1.4.8 (build date 2017-05-11 18:25:06+0100)
ng
1494527604: Config loaded from /etc/mosquitto/mosquitto.conf.
1494527604: Opening websockets listen socket on port 9001.
1494527604: Opening ipv4 listen socket on port 1883.
1494527604: Opening ipv6 listen socket on port 1883.
```