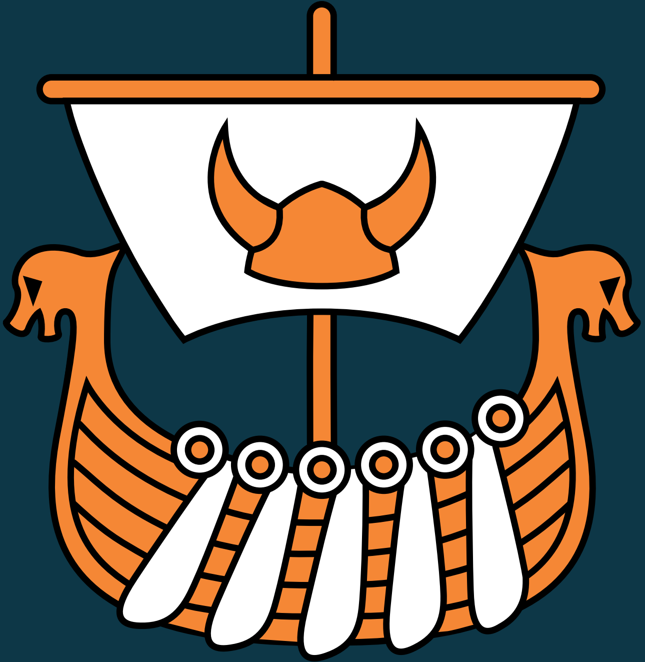


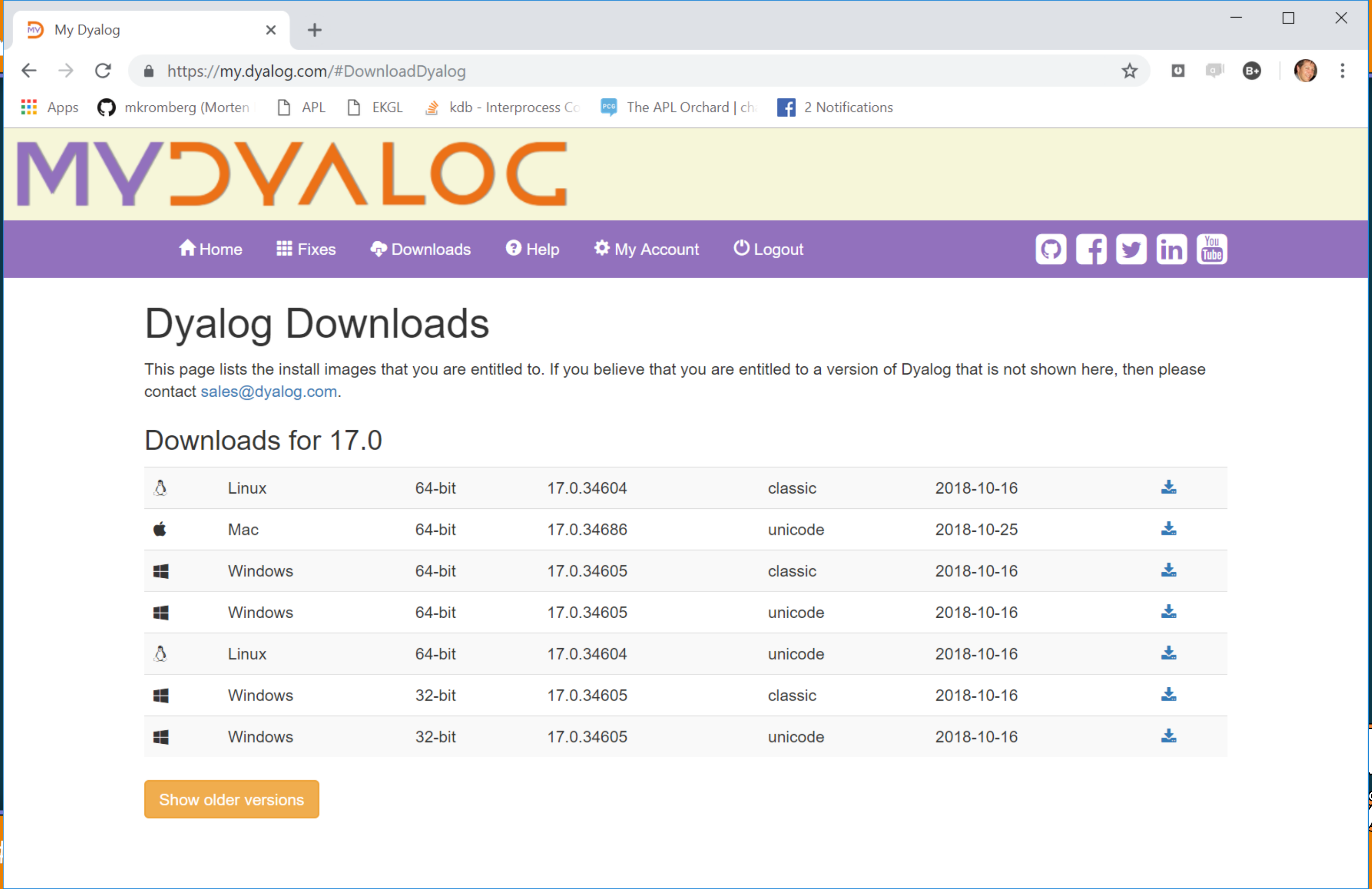
DYALOG

Belfast 2018

Workshop TP2 APL in the Cloud APL, RIDE & JSONServer

Morten Kromberg, CXO, Dyalog
Andy Shiers, COO, Dyalog

Installing Dyalog APL 17.0



Dyalog Downloads

This page lists the install images that you are entitled to. If you believe that you are entitled to a version of Dyalog that is not shown here, then please contact sales@dyalog.com.

Downloads for 17.0

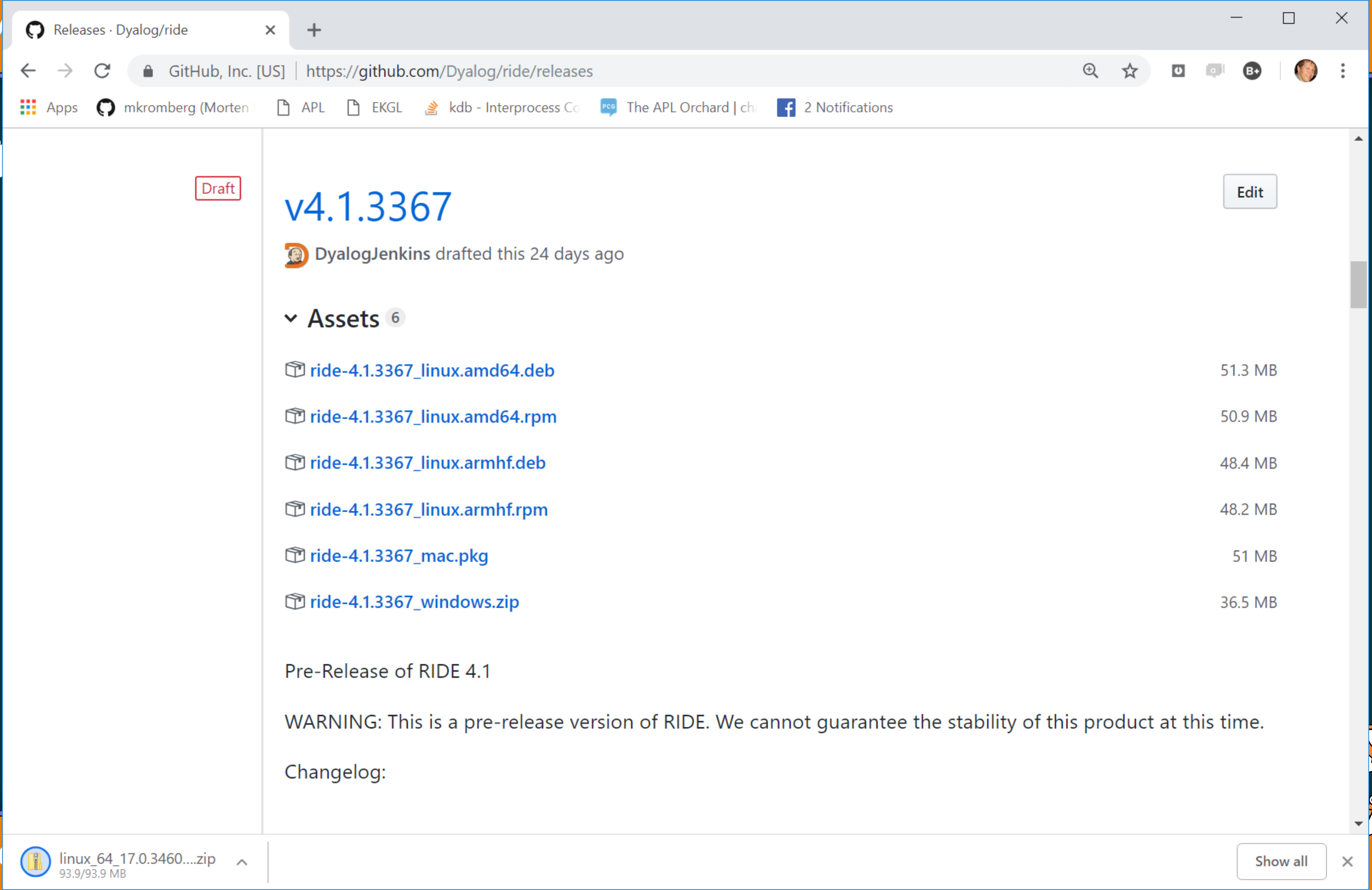
Linux	64-bit	17.0.34604	classic	2018-10-16	⬇
Mac	64-bit	17.0.34686	unicode	2018-10-25	⬇
Windows	64-bit	17.0.34605	classic	2018-10-16	⬇
Windows	64-bit	17.0.34605	unicode	2018-10-16	⬇
Linux	64-bit	17.0.34604	unicode	2018-10-16	⬇
Windows	32-bit	17.0.34605	classic	2018-10-16	⬇
Windows	32-bit	17.0.34605	unicode	2018-10-16	⬇

Show older versions

Installing APL

```
dpkg -i linux_64_17.0.34604_unicode.x86_64.deb
```

Installing RIDE 4.1



v4.1.3367

DyalogJenkins drafted this 24 days ago

Assets 6

ride-4.1.3367_linux.amd64.deb	51.3 MB
ride-4.1.3367_linux.amd64.rpm	50.9 MB
ride-4.1.3367_linux.armhf.deb	48.4 MB
ride-4.1.3367_linux.armhf.rpm	48.2 MB
ride-4.1.3367_mac.pkg	51 MB
ride-4.1.3367_windows.zip	36.5 MB

Pre-Release of RIDE 4.1

WARNING: This is a pre-release version of RIDE. We cannot guarantee the stability of this product at this time.

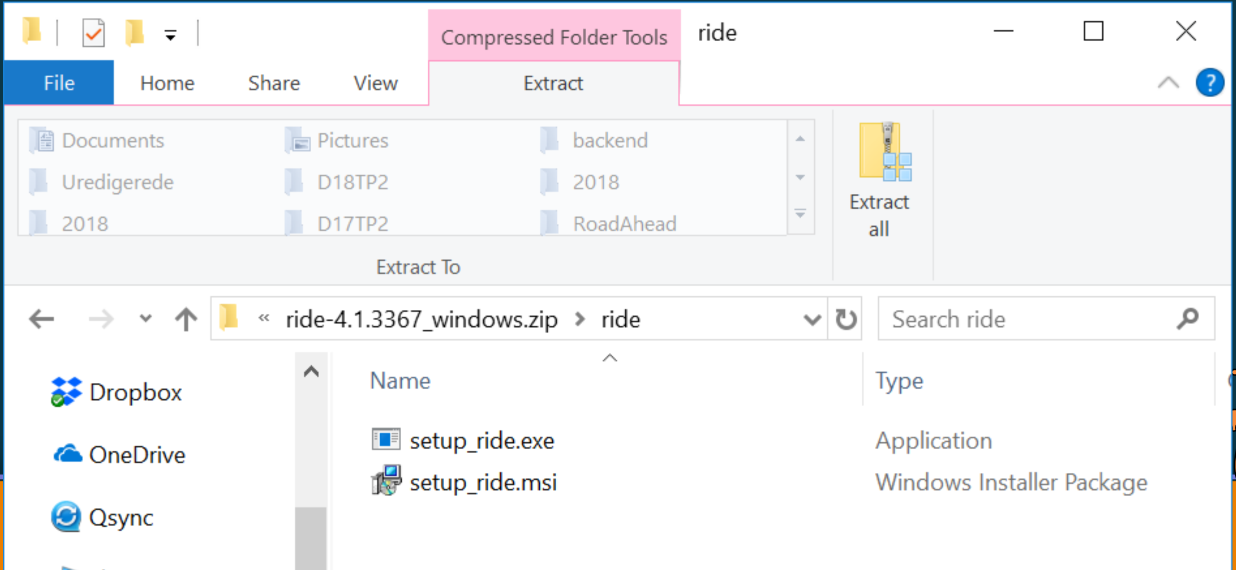
Changelog:

Installing RIDE

- Linux:

```
dpkg –i ride-4.1.3367_linux.amd64.deb
```

- Windows:



Name	Type
setup_ride.exe	Application
setup_ride.msi	Windows Installer Package

Secure Shell (ssh)

- *ssh* is a widely used protocol for making executing commands on a remote computer
- It is always secure (encrypted) even if you log in with a userid and password
- It supports the use of key pairs to validate login without a password
  - You log in with a user id and a private key
  - The ssh client and server negotiate, and if the public key corresponding to your private key exists in the right place, you are granted access
  - The right place is typically the file `/home/user/.ssh/authorized_keys` Which contains concatenated public keys

ssh setup

- ssh (secure shell) is the safest way to connect to a Linux machine.
- If you are going to connect to your machine from Windows, follow these instructions:

- First, install openssh server if necessary

```
sudo apt-get install openssh-server
```

- ssh relies on a key pair, which we will generate

Generate a Key Pair

- Create / verify the existence of a directory called $HOME/.ssh to store the keys.
- Run the ssh-keygen command to generate public and private keys:

```
ssh-keygen -t  rsa
```

- This creates the following files in the $HOME/.ssh directory:
  - Private key: id_rsa
  - Public key: id_rsa.pub

Install public key on server

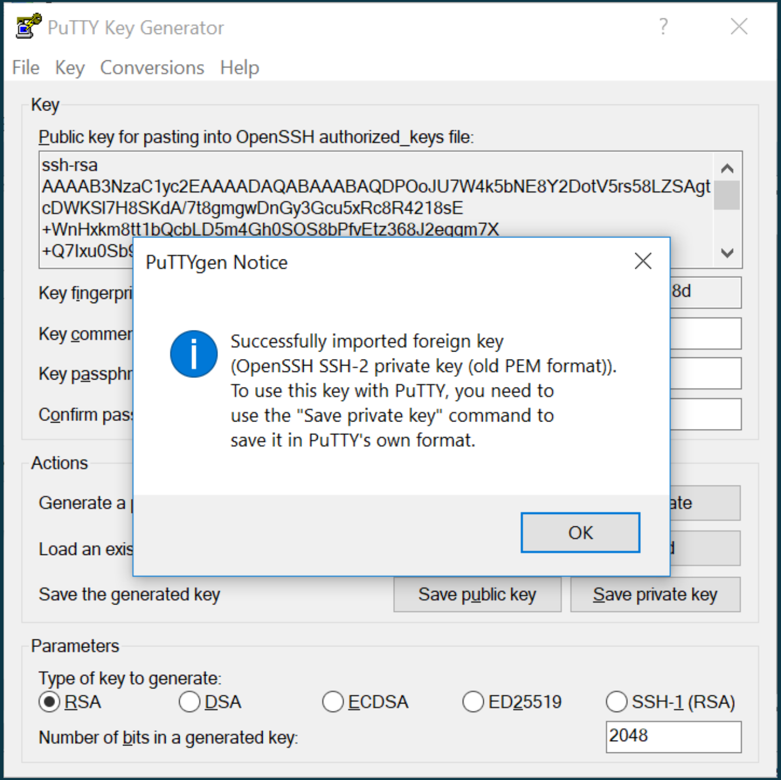
- Append the public key to the authorized_keys file on the Linux machine:

```
cd ~/.ssh
cat id_rsa.pub >> authorized_keys
```

- This will now allow ssh from a client which is able to present the private key file (id_rsa)

Install public key on client

- Copy the private key file to the client and give it a good name like `mary.key`
- If you are going to use PuTTY, you need to convert it to .ppk format with PuTTYGen:
  - Load the .key file and save it as .ppk



PuTTY Key Generator

File Key Conversions Help

Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDPOoJU7W4k5bNE8Y2DotV5rs58LZSAgt
cDWKSI7H8SKdA/7t8gmgwDnGy3Gcu5xRc8R4218sE
+WnHxkm8tt1bQcbLD5m4Gh0SOS8bPfvEtz368J2egqm7X
+Q7Ixu0Sb
```

PuTTYgen Notice

Successfully imported foreign key (OpenSSH SSH-2 private key (old PEM format)). To use this key with PuTTY, you need to use the "Save private key" command to save it in PuTTY's own format.

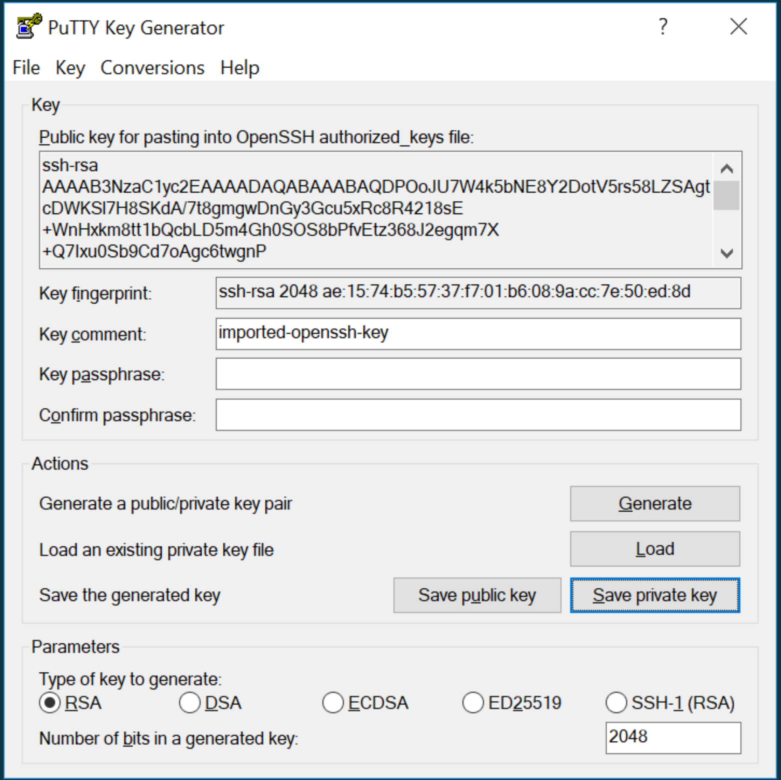
OK

Save the generated key — Save public key — Save private key

Parameters

Type of key to generate: [✓ RSA] DSA ECDSA ED25519 SSH-1 (RSA)

Number of bits in a generated key: 2048



PuTTY Key Generator

File Key Conversions Help

Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDPOoJU7W4k5bNE8Y2DotV5rs58LZSAgt
cDWKSI7H8SKdA/7t8gmgwDnGy3Gcu5xRc8R4218sE
+WnHxkm8tt1bQcbLD5m4Gh0SOS8bPfvEtz368J2egqm7X
+Q7Ixu0Sb9Cd7oAgc6twgnP
```

Key fingerprint:	ssh-rsa 2048 ae:15:74:b5:57:37:f7:01:b6:08:9a:cc:7e:50:ed:8d
Key comment:	imported-openssh-key
Key passphrase:	
Confirm passphrase:	

Actions

Generate a public/private key pair — Generate

Load an existing private key file — Load

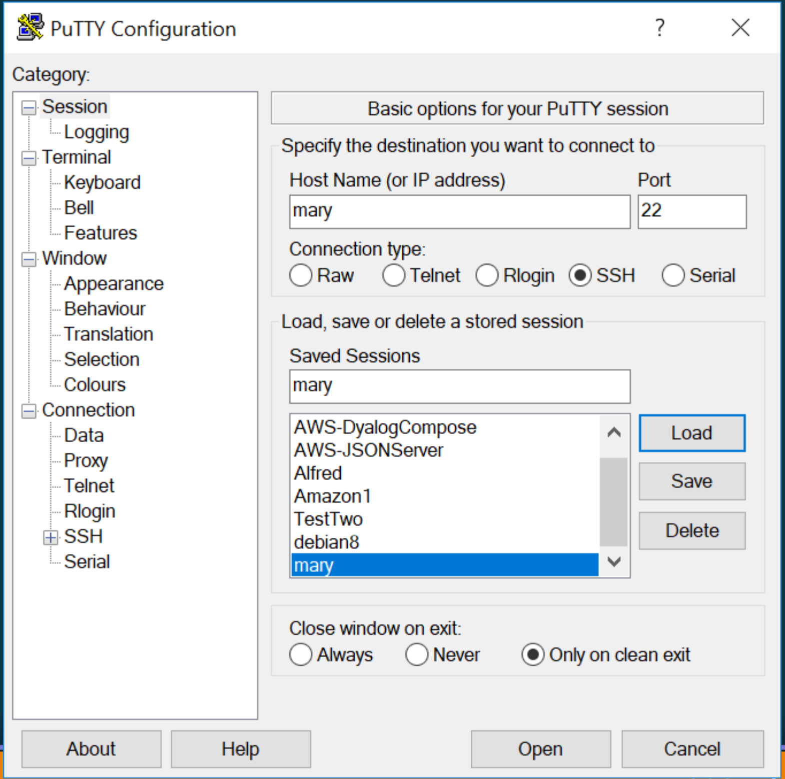
Save the generated key — Save public key — Save private key

Parameters

Type of key to generate: [✓ RSA] DSA ECDSA ED25519 SSH-1 (RSA)

Number of bits in a generated key: 2048

Connect with PuTTY



PuTTY Configuration

Category:

- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data
  - Proxy
  - Telnet
  - Rlogin
  - SSH
  - Serial

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): mary

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

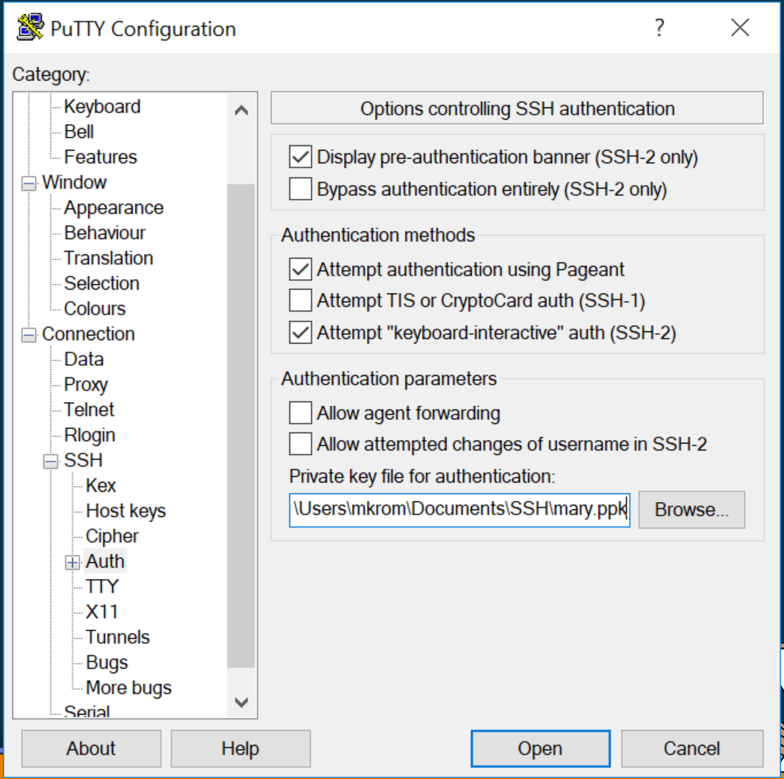
Saved Sessions: mary

AWS-DyalogCompose
AWS-JSONServer
Alfred
Amazon1
TestTwo
debian8
mary

Load Save Delete

Close window on exit: Always Never [✓ Only on clean exit]

About Help Open Cancel



PuTTY Configuration

Category:

- Keyboard
- Bell
- Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data
  - Proxy
  - Telnet
  - Rlogin
  - SSH
    - Kex
    - Host keys
    - Cipher
    - Auth
    - TTY
    - X11
    - Tunnels
    - Bugs
    - More bugs
  - Serial

Options controlling SSH authentication

- [✓] Display pre-authentication banner (SSH-2 only)
- [ ] Bypass authentication entirely (SSH-2 only)

Authentication methods

- [✓] Attempt authentication using Pageant
- [ ] Attempt TIS or CryptoCard auth (SSH-1)
- [✓] Attempt "keyboard-interactive" auth (SSH-2)

Authentication parameters

- [ ] Allow agent forwarding
- [ ] Allow attempted changes of username in SSH-2

Private key file for authentication: \Users\mkrom\Documents\SSH\mary.ppk

Browse...

About Help Open Cancel

Exercise 1

- Install Dyalog APL under Linux
- Install RIDE under Windows or Linux
- Use RIDE to start an APL Session on your Linux machine
- Create a folder to contain a simple application with one or two functions that you will turn into a service
  - The functions should take JSON-able data
  - Experiment with ⎕JSON to see what a suitable argument will look like in JSON format, and note that down (you will need it in the next exercise).
- Use]save to populate the folder

JSONServer

JSONServer

- A TCP Server based on Conga

JSONServer

- A TCP Server based on Conga
- Uses ⎕JSON to convert incoming data to APL arrays

```
POST /GetSign HTTP 1.1
[10,31]
```

JSONServer

- A TCP Server based on Conga
- Uses ⎕JSON to convert incoming data to APL arrays
- Calls Function

```
POST /GetSign HTTP 1.1
[10,31]
```

```
r←GetSign 10 31
```

JSONServer

- A TCP Server based on Conga
- Uses ⎕JSON to convert incoming data to APL arrays
- Calls Function
- Converts results back to JSON and returns HTTP

```
POST /GetSign HTTP 1.1
[10,31]
```

```
r←GetSign 10 31
```

```
HTTP/1.1 200 OK
"Scorpio"
```

JSONServer Features

JSONServer Features

- Can Serve Up

JSONServer Features

- Can Serve Up
  - Functions in a namespace (including #)
    - The `AllowedFns` property can be used to control which functions to expose

JSONServer Features

- Can Serve Up
  - Functions in a namespace (including #)
    - The `AllowedFns` property can be used to control which functions to expose
  - A folder full of .dyalog files

JSONServer Features

- Can Serve Up
  - Functions in a namespace (including #)
    - The `AllowedFns` property can be used to control which functions to expose
  - A folder full of .dyalog files
  - Nested folders / namespaces
    - URLs a la localhost:8080/ns/foo

JSONServer Features

- Can Serve Up
  - Functions in a namespace (including #)
    - The `AllowedFns` property can be used to control which functions to expose
  - A folder full of .dyalog files
  - Nested folders / namespaces
    - URLs a la localhost:8080/ns/foo
- Uses `⎕JSON` to convert incoming data & results to or from APL arrays

JSONServer Features

- Can Serve Up
  - Functions in a namespace (including #)
    - The `AllowedFns` property can be used to control which functions to expose
  - A folder full of .dyalog files
  - Nested folders / namespaces
    - URLs a la localhost:8080/ns/foo
- Uses `⎕JSON` to convert incoming data & results to or from APL arrays
- Can be started from the command line

JSONServer Features

- Can Serve Up
  - Functions in a namespace (including #)
    - The `AllowedFns` property can be used to control which functions to expose
  - A folder full of .dyalog files
  - Nested folders / namespaces
    - URLs a la localhost:8080/ns/foo
- Uses `⎕JSON` to convert incoming data & results to or from APL arrays
- Can be started from the command line

Get it from https://github.com/Dyalog/JSONServer

Exercise 2

- Install JSONServer:
  git clone https://github.com/Dyalog/JSONServer

- Start APL and]load your functions from Exercise 1 into a namespace, for example:

```
 )NS MyNs
 ]load /app-folder/* -target=MyNs
```

- Verify that your functions were loaded.

Exercise 2 - Continued

- Start JSONServer

```
]load /Devt/JSONServer/Source/JSONServer
srv←⎕NEW JSONServer
)ns Zodiac
]load C:\D18TP2\ZodiacService\backend\* -target=Zodiac
srv.CodeLocation←#.Zodiac
srv.Port←8080
srv.Start
```

- Test it using browser to localhost:8080 or curl (see below)

```
srv.Stop
```

- CURL:
  curl --header "content-type: application/json"
   --data "JSON Argument" http://127.0.0.1:8080/YourFunction