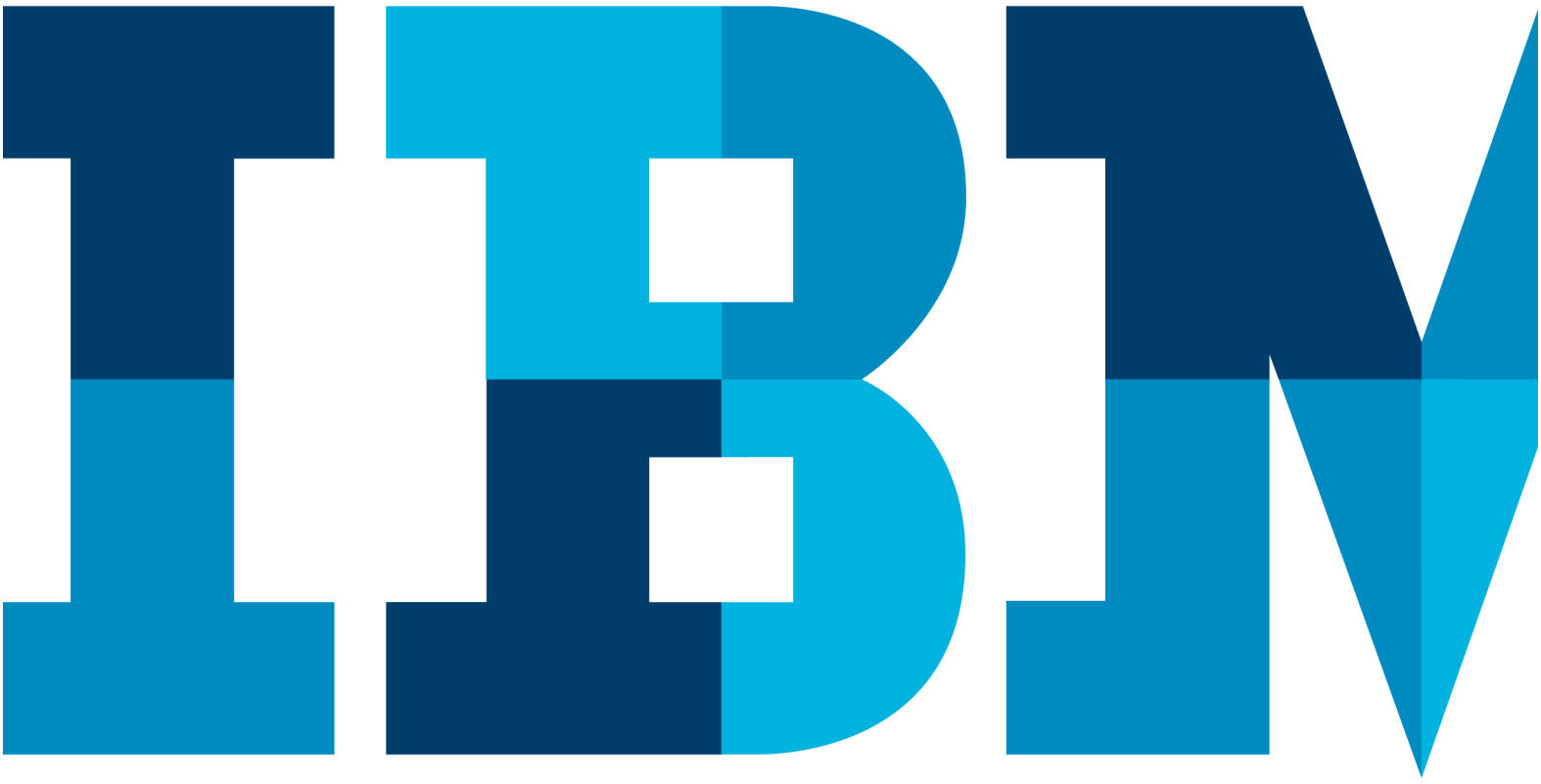
IBM Blockchain Hands-On
Blockchain Unchained

Lab Three – VM – Exercises



IBM

Contents

Overview

The aim of this lab is to introduce you to chaincode development by showing how to deploy your first chaincode to a new blockchain service in a virtual machine environment.

We will use a sample piece of chaincode (Smart Contract) called **example02** as the foundation for our lab. This sample is provided as part of the Hyperledger Fabric code.

Once deployed, the chaincode can be invoked and queried.

Introduction

Pre-requisites:

- 4 cores
- 4GB RAM
- VMWare V10+
- The lab virtual machine

The virtual machine is based on Linux Ubuntu 14.04 and contains Hyperledger Fabric V0.6, Golang, Git, Vagrant, Visual Studio Code with the “Encode Decode” extension and Firefox.

A network needs to be visible to the virtual machine (even if the network is just to the host environment). If you do not see the up/down arrows in the status bar at the top of the screen, or if you receive errors about no network being available, please tell the lab leader. The virtual machine might need to be reconfigured in NAT mode.

There are no additional files or software that is proprietary to the lab in the virtual machine. This means that the lab may be run on a machine without the without a lab virtual machine if Hyperledger Fabric and the other pre-requisites have been installed.

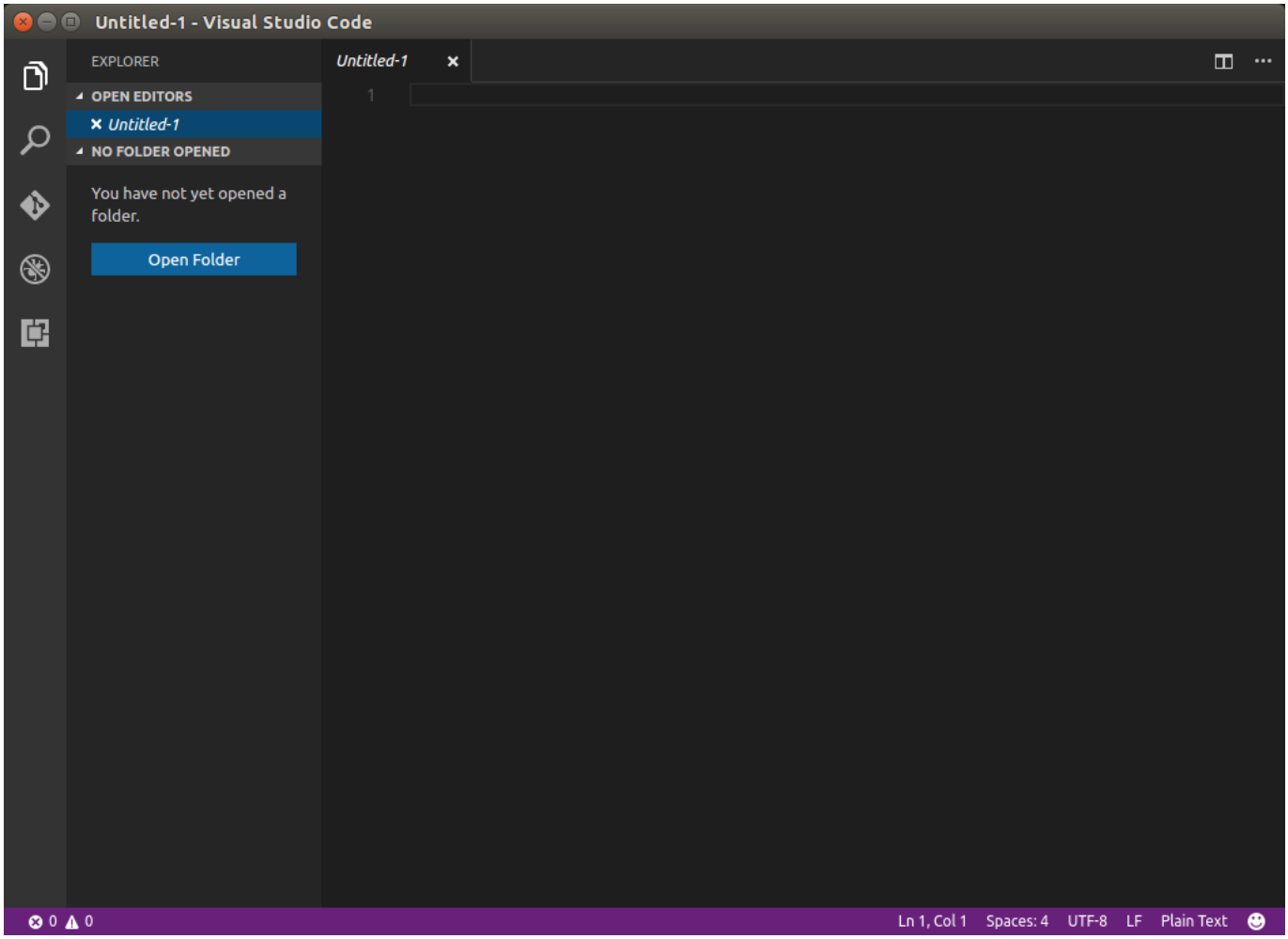
It is recommended that students have previously completed the Blockchain Explained and Blockchain Explored labs.

Section 1. Reviewing the chaincode

We will first locate the source of the example02 chaincode and review it.

__1. To start the Visual Studio Code IDE, **left-click** the icon:

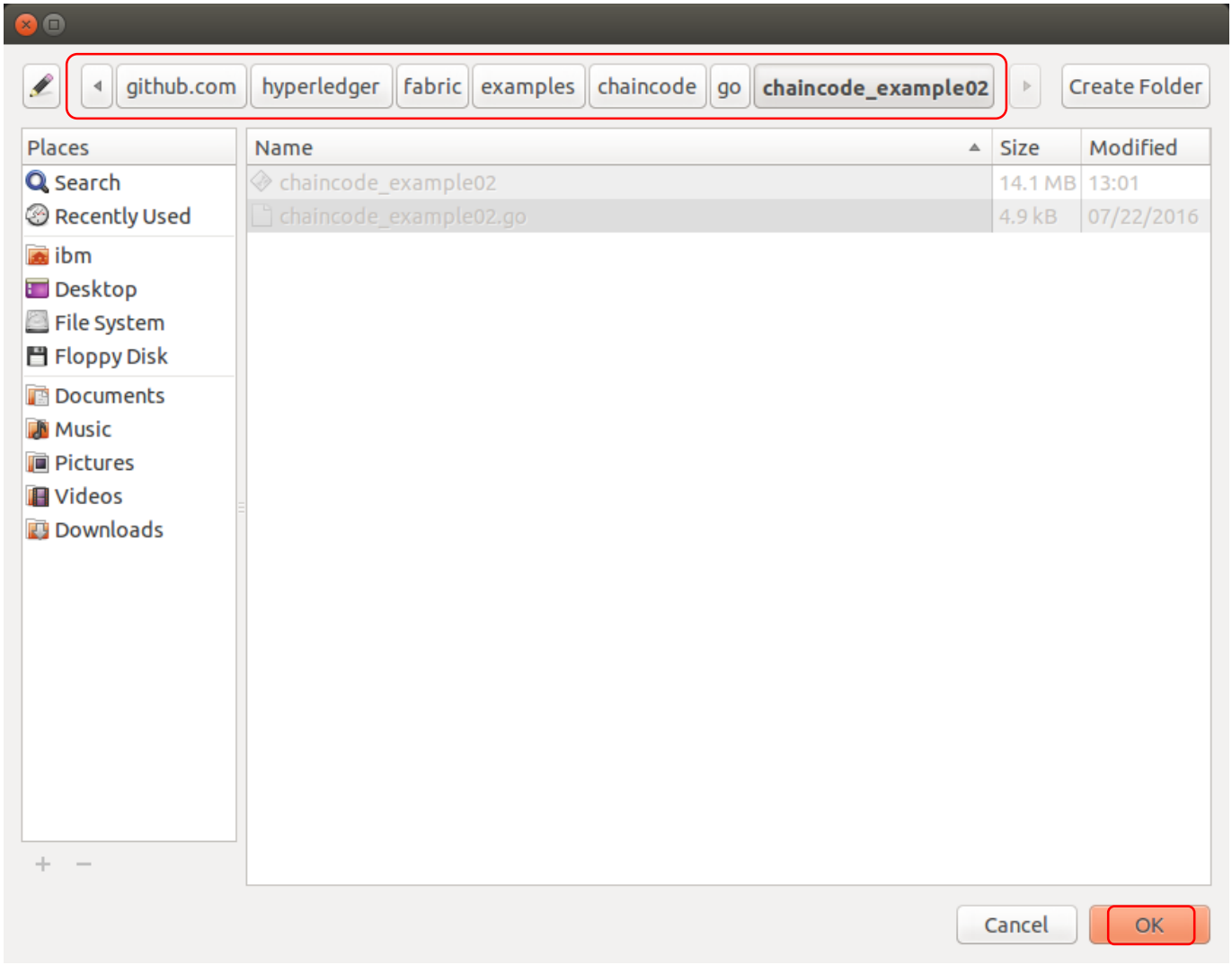
__2. The Visual Studio Code editor is loaded.



__3. The **example02** chaincode can be found in the following directory:

```
/home/ibm/Documents/IBM_Blockchain/Projects/src/github.com/hyperledger/fabric
/examples/chaincode/go/chaincode_example02
```

Within the editor, select **File > Open Folder** from the menu and navigate to directory above, and select **OK**.



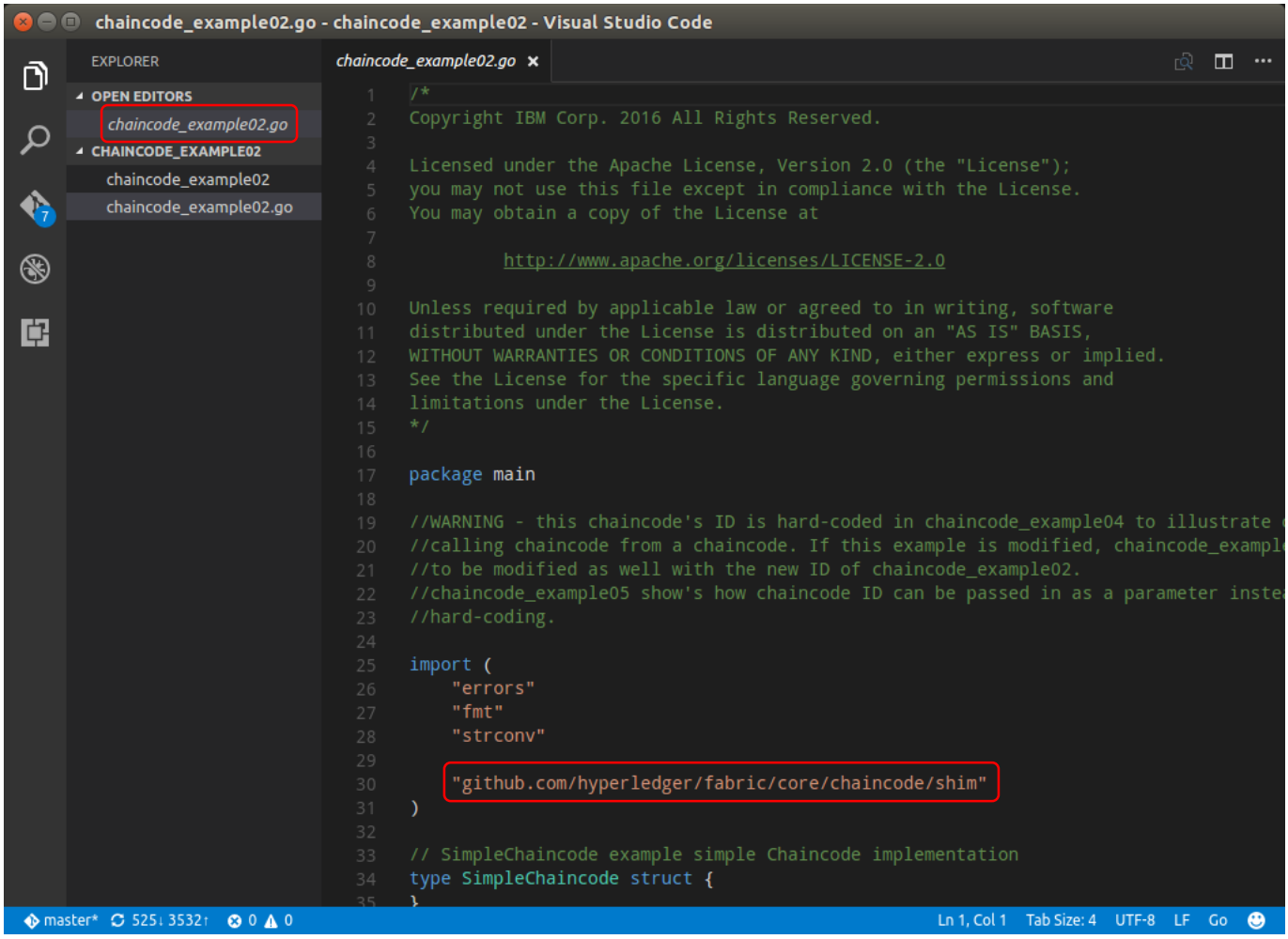
__4. Using the Explorer pane to the left of the editor, **left-click** the file **chaincode_example02** (source file). This will open the file in the editor window to view.

Every chaincode deployed to Hyperledger Fabric must adhere to the shim (githum.com/hyperledger/fabric/core/chaincode/shim) which defines the interface.

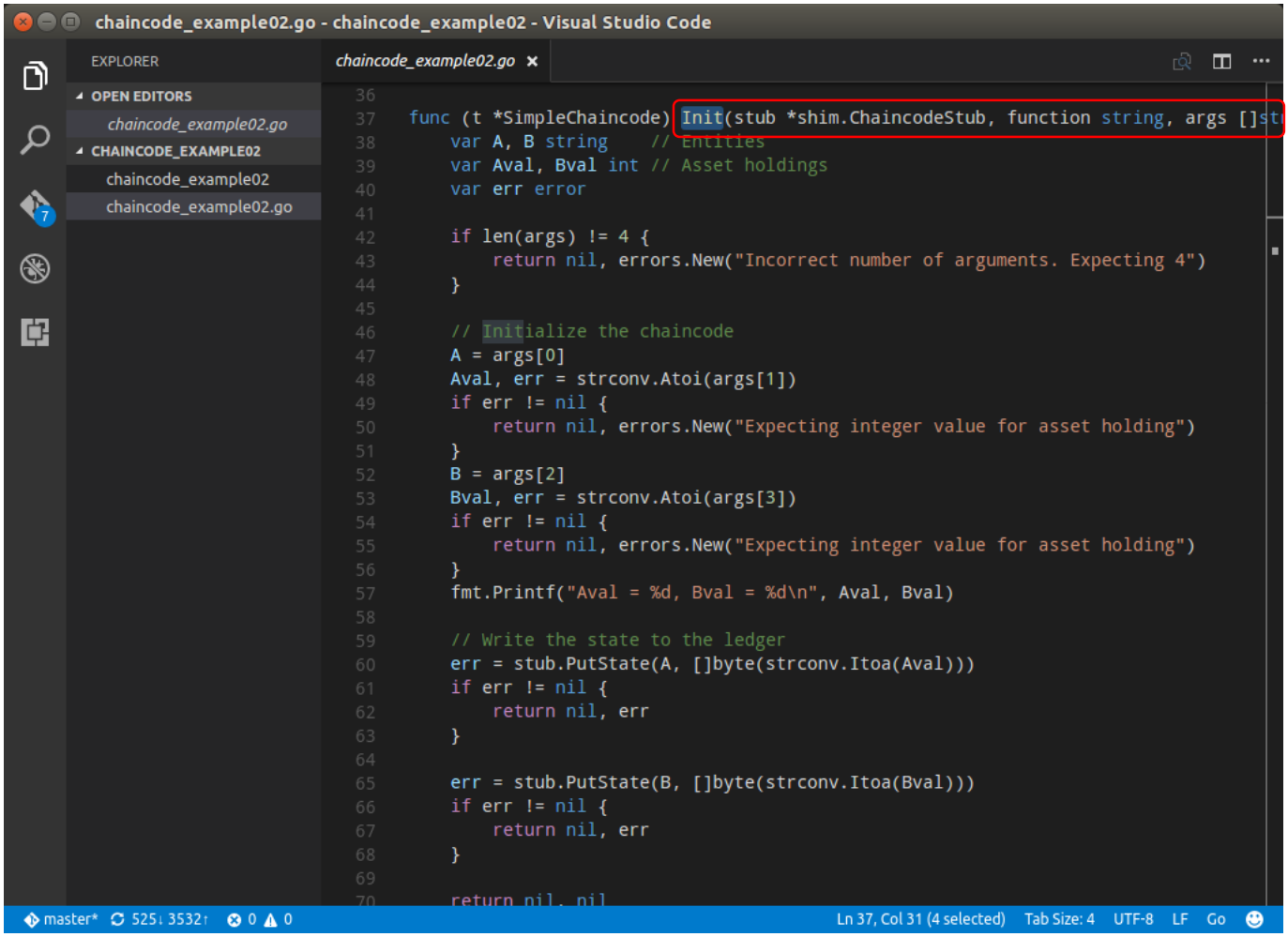
Upon initialization, example02 creates two key/value pairs in the world-state. Each key is an identifier string (e.g. “A”) whose value is an associated integer balance (e.g. 100). Transactions that invoke this chaincode will increment the balance of one identifier while decrementing the other.

For example:

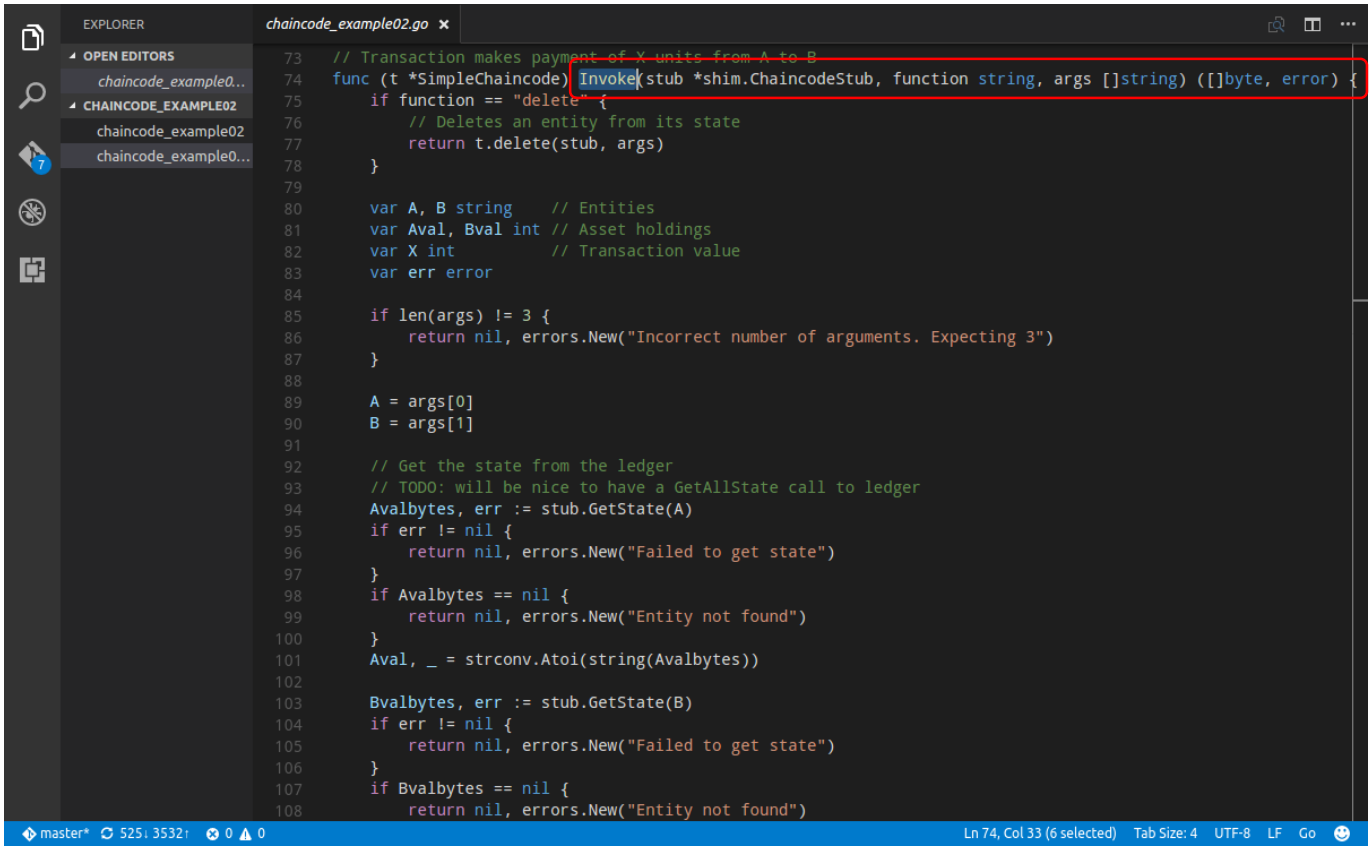
> **Initializing** the chaincode with [“a”, “100”, “b”, “200”] will set up “a” to be “100” and “b” to be “200”.
> **Invoking** a transaction with [“a”, “b”, “10”] will decrement the value of “a” by 10 and increment “b” by the same amount.
> **Querying** the value of “a” and “b” at this point will yield “a” to have the value “90” and “b” to have “210”.



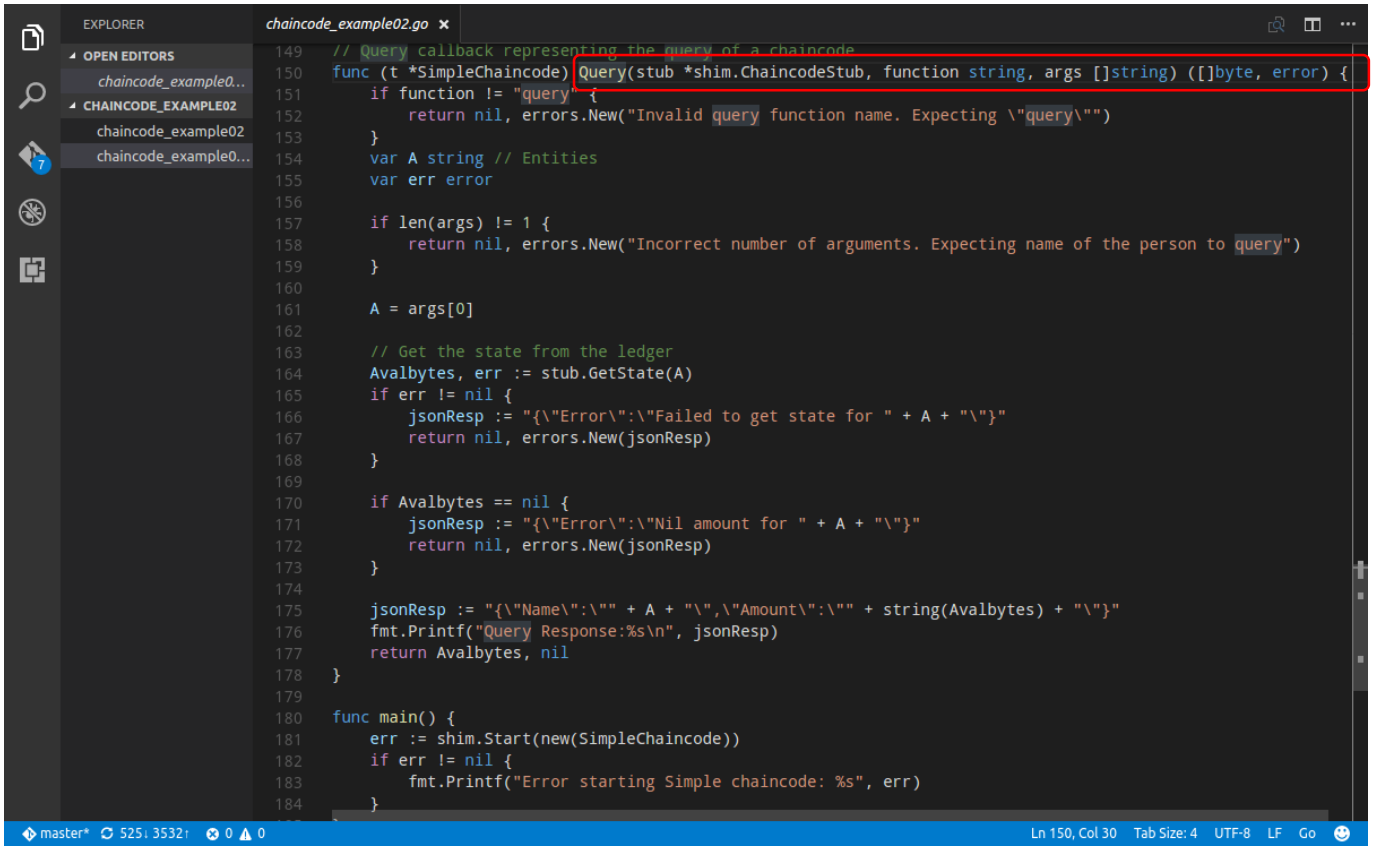
__5. Review the **`Init()`** method of the chaincode. The **`Init()`** method is called once on deploying the chaincode. The deployment of the chaincode will be recorded as a transaction in the blockchain.



__6. Review the **`Invoke()`** method of the chaincode. This is called everytime the chaincode is "invoked". The call to **`Invoke()`** will be recorded as a transaction in the blockchain. An invoke normally updates content of the world-state.



__7. Review the **`Query()`** method of the chaincode. This is called everytime the chaincode is "queried". The call to **`Query()`** will not be recorded as a transaction in the blockchain. A query is a read-only transaction and therefore cannot update content in the world-state.



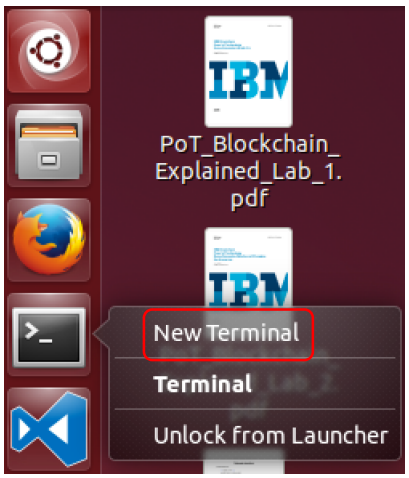
We will now deploy and run this chaincode.

Section 2. Accessing the Vagrant Environment

Vagrant is a virtual OS *inside* the lab virtual machine that is used to compile and run the Hyperledger Fabric. Having a nested virtualisation environment helps ensure consistency in the runtime environment, allowing you to see similar compilation and runtime results whether you are developing on MacOS, Windows, Linux or whatever. The Vagrant VM is based on Ubuntu Linux.

In this section you will create a shell command prompt in Vagrant which is where the Hyperledger Fabric peers are running. Once in this environment you will be able to issue **peer** commands to deploy chaincode (Smart Contracts).

__1. Start a new terminal window. **Right-click** on the terminal icon, and select "**New Terminal**".



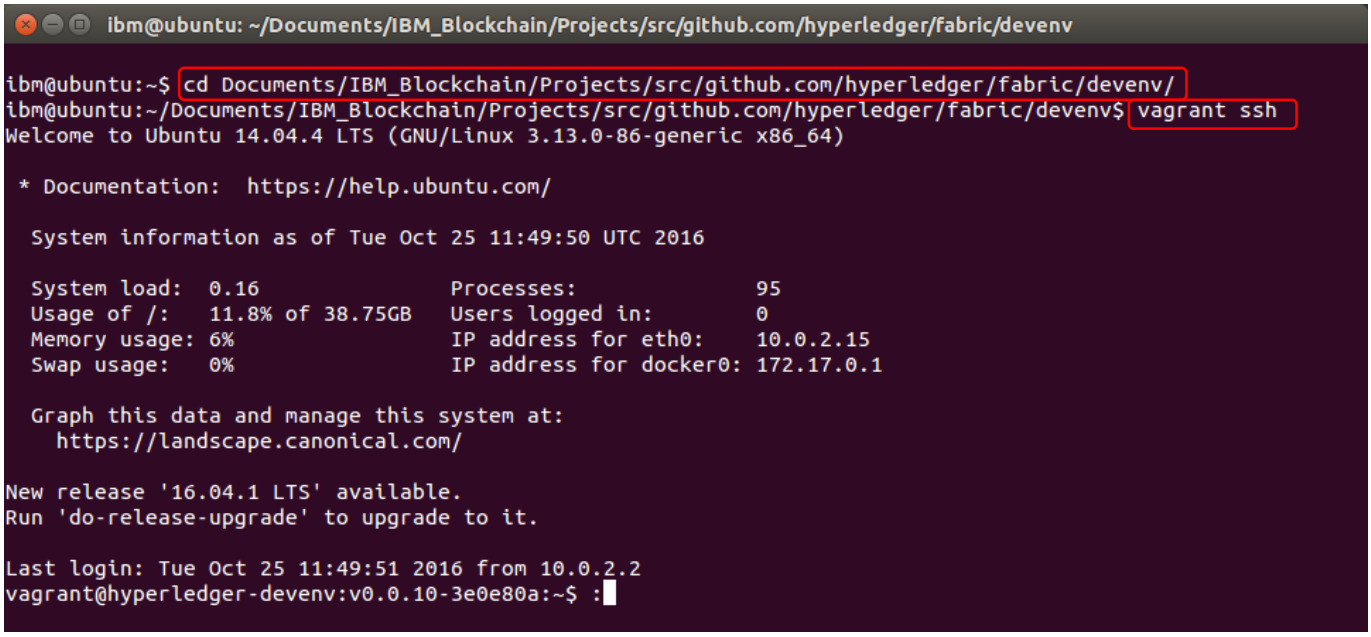
__2. As Hyperledger Fabric is running in a Vagrant environment, create a shell command prompt.

Vagrant has been started automatically in the Ubuntu VM. In order to run the fabric **peer** commands, a Vagrant shell command prompt needs to be created.

From the command line in the new terminal, change to the following directory directory which has the Vagrant image file, by entering:

```
cd Documents/IBM_Blockchain/Projects/src/github.com/hyperledger/fabric/devenv
```

__3. Start the Vagrant shell by entering: **`vagrant ssh`**


ibm@ubuntu: ~/Documents/IBM_Blockchain/Projects/src/github.com/hyperledger/fabric/devenv
ibm@ubuntu:~$ cd Documents/IBM_Blockchain/Projects/src/github.com/hyperledger/fabric/devenv/
ibm@ubuntu:~/Documents/IBM_Blockchain/Projects/src/github.com/hyperledger/fabric/devenv$ vagrant ssh
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.13.0-86-generic x86_64)
* Documentation: https://help.ubuntu.com/
System information as of Tue Oct 25 11:49:50 UTC 2016
System load: 0.16 Processes: 95
Usage of /: 11.8% of 38.75GB Users logged in: 0
Memory usage: 6% IP address for eth0: 10.0.2.15
Swap usage: 0% IP address for docker0: 172.17.0.1
Graph this data and manage this system at:
https://landscape.canonical.com/
New release '16.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
Last login: Tue Oct 25 11:49:51 2016 from 10.0.2.2
vagrant@hyperledger-devenv:v0.0.10-3e0e80a:~$:

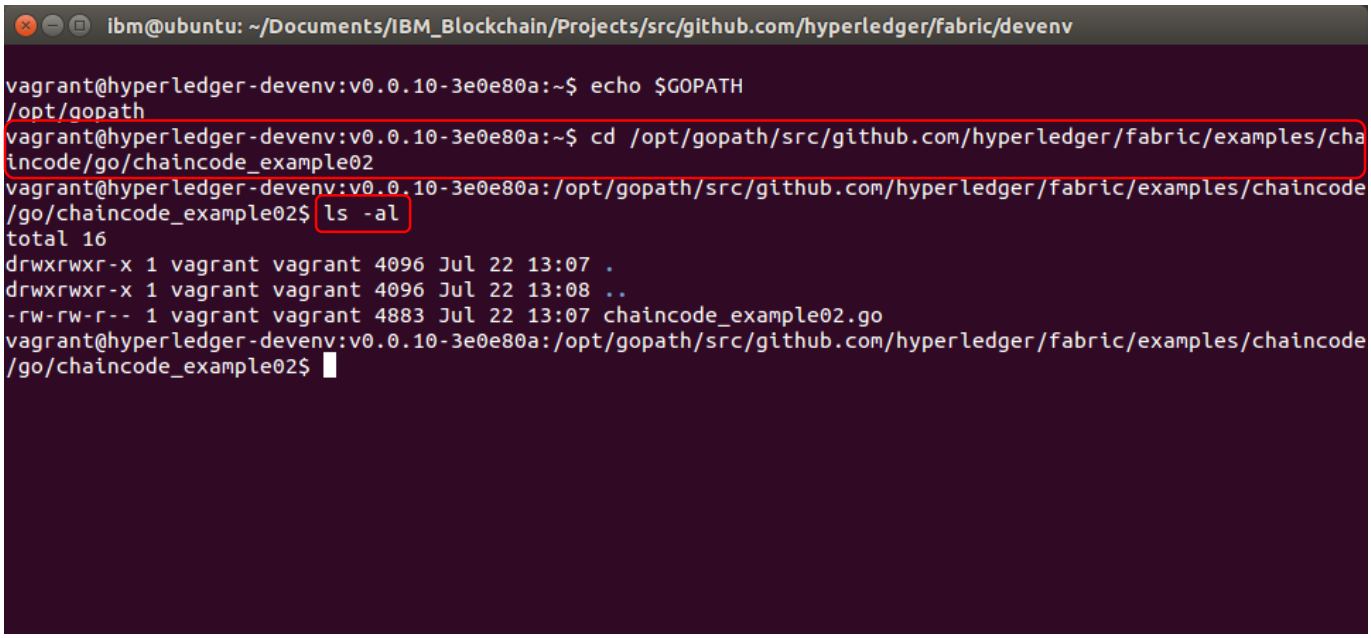
Section 3. Deploying the Chaincode

We will now deploy the chaincode from within the Vagrant environment previously started.

__1. Locate the example chaincode to deploy

The environment **$GOPATH** locates the root of the Hyperledger Fabric code, and chaincode must exist as a subdirectory of this root to deploy using the **peer** command. Change the directory to locate the **example02** chaincode, and list the contents of the directory by entering:

```
cd $GOPATH/src/github.com/hyperledger/fabric/examples/chaincode/go/chaincode_example02
ls -al
```

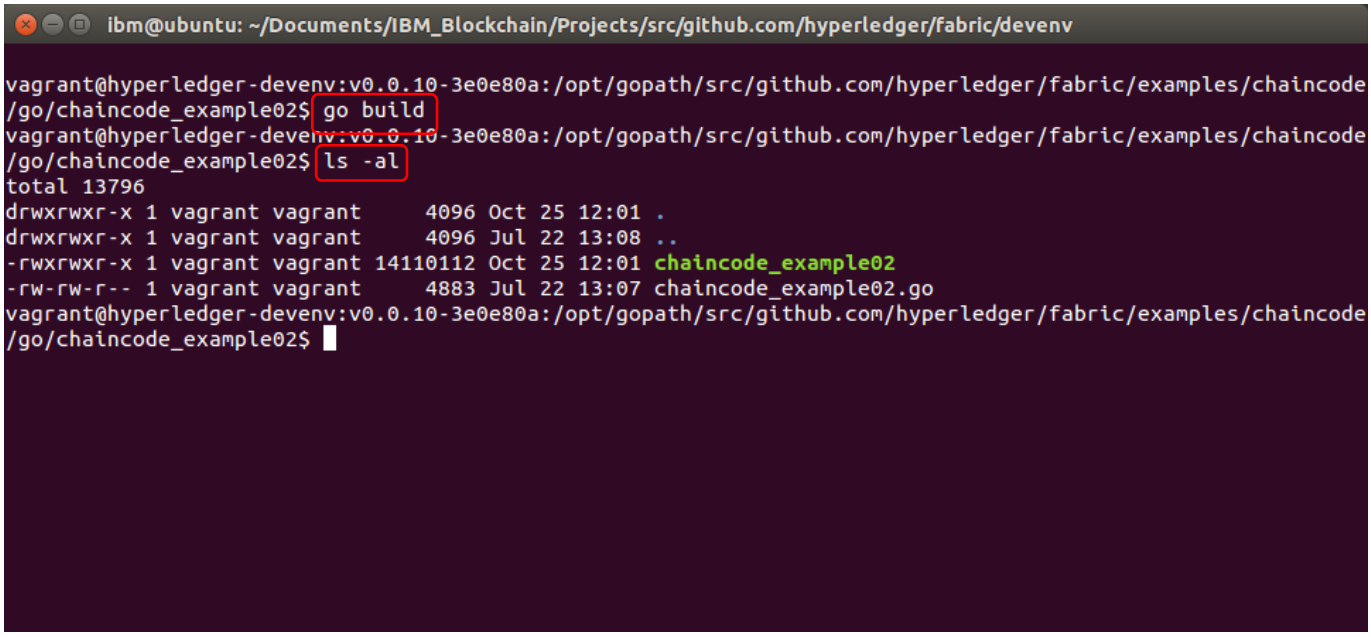


__2. Compile example02.

It is good practice to compile chaincode before deployment. This is not strictly necessary as the chaincode source is deployed to the blockchain, but doing so helps eliminate any errors as early as possible.
The environment is set up to include all pre-requisites for building chaincode (Hyperledger Fabric, shim, GOLang compiler, RocksDB etc).

Build **example02** by entering the following commands (there should be no errors):

```
go build
ls -al
```



Note: The output is the compiled file **chaincode_example02**.

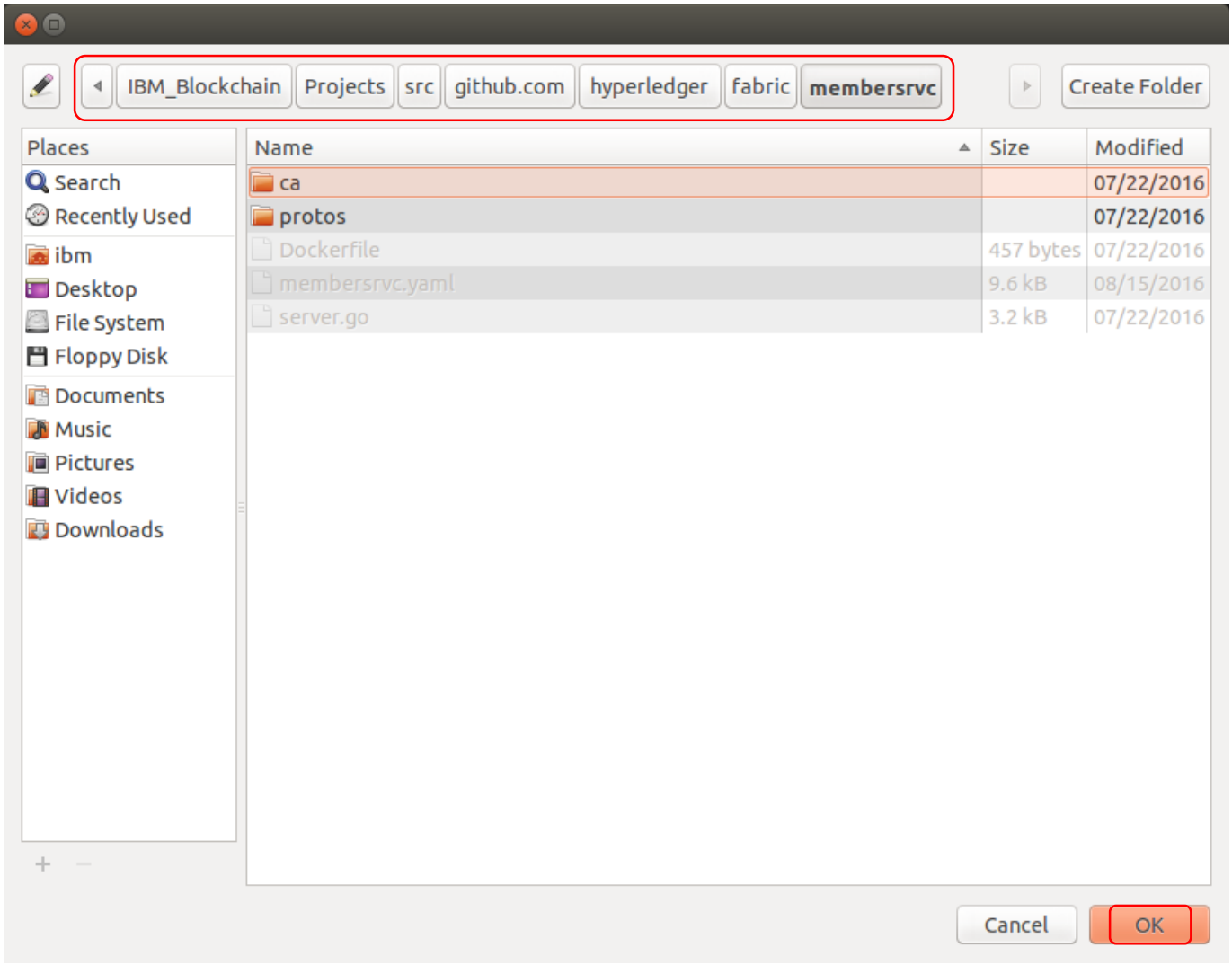
Before the chaincode can be deployed, the user (Administrator) must first login to the Fabric peer process, which is already running.

A set of users are configured, and these can be found in the **membersrvc.yaml** file.

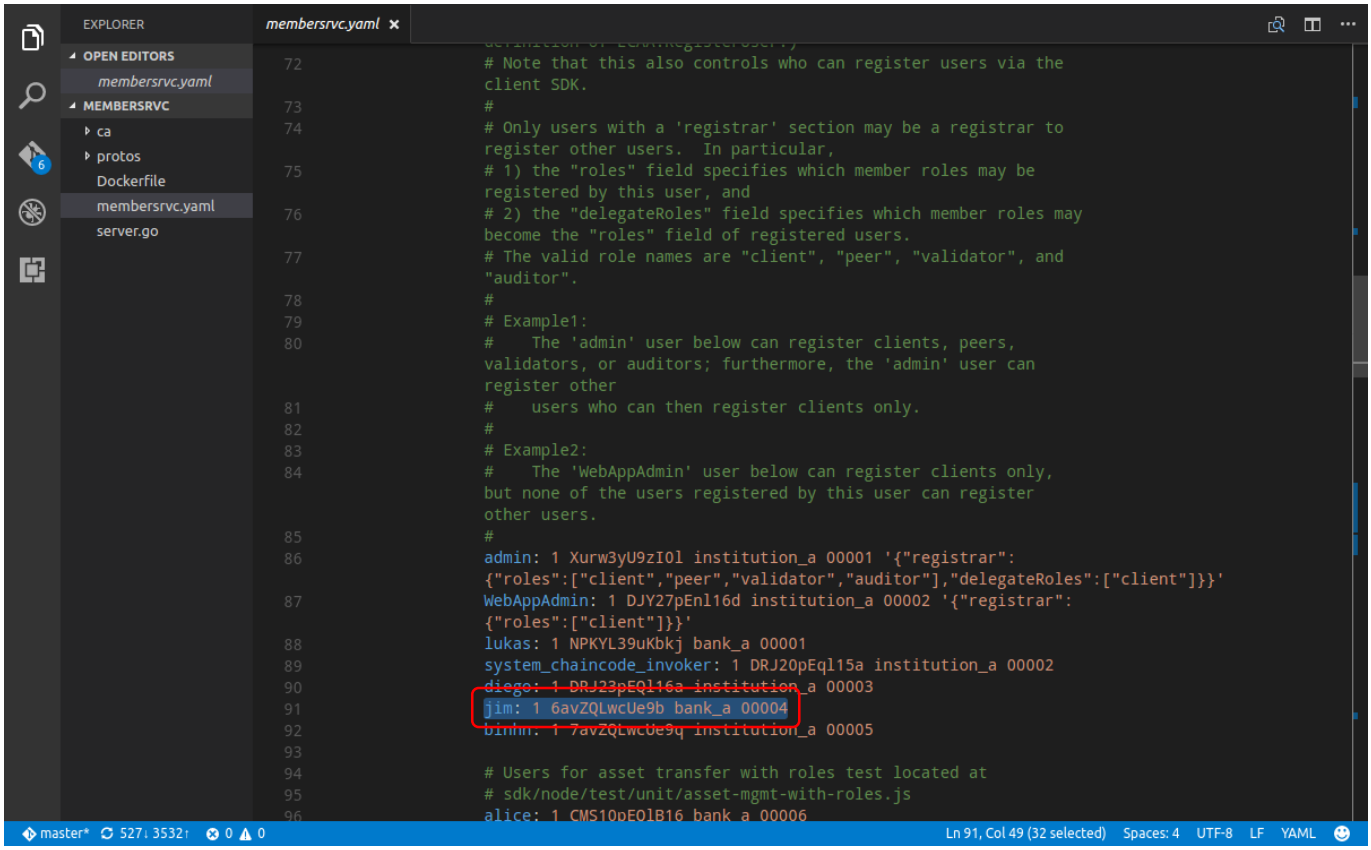
__3. Open the **membersrvc.yaml** file in VSCode, located in:

`/home/ibm/Documents/IBM_Blockchain/Projects/src/github.com/hyperledger/fabric/membersrvc`

Within the Visual Studio Code editor, select **File > Open Folder** and navigate to the above directory, and select **OK**.

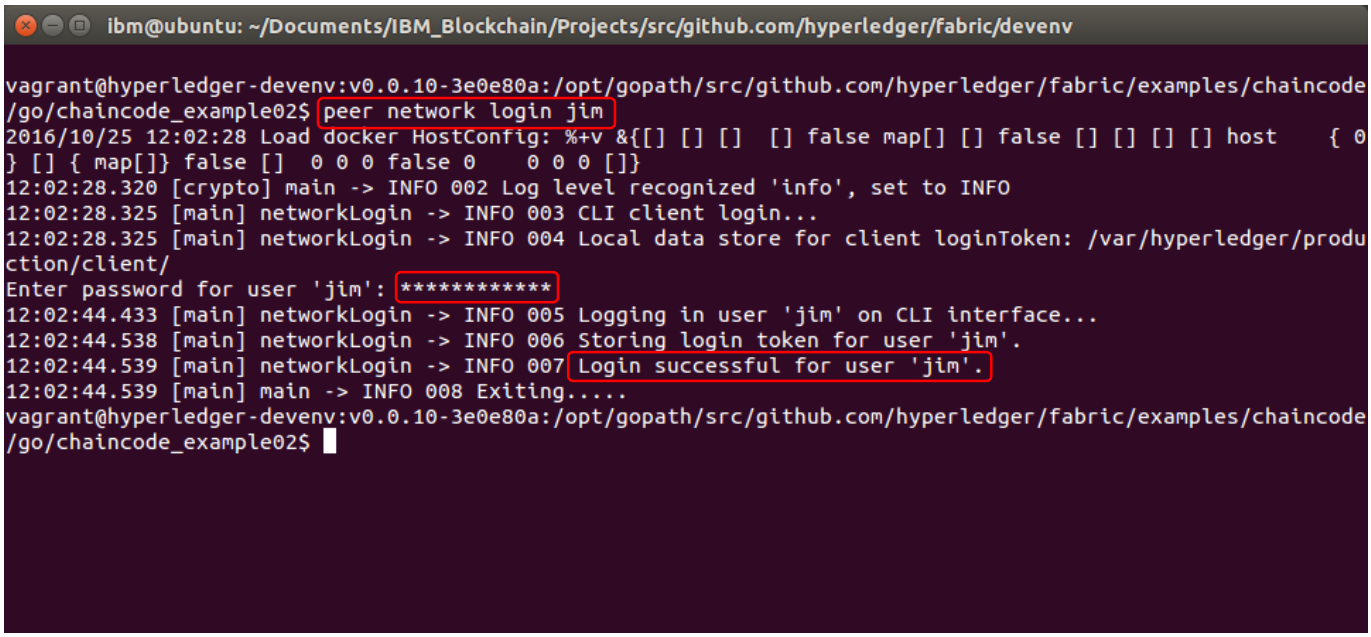


__4. Using the Explorer pane to the left of the editor, **left-click** the file **membersrvc.yaml**. This will open the file in the editor window. Scroll down to review user information. For the purposes of this lab we will use **jim** with the password **6avZQLwcUe9b.**



__5. Login user **jim** on the peer. Switch back to the terminal window, and enter the following command (Note: The password ***6avZQLwcUe9b*** is entered separately):

```
peer network login jim
6avZQLwcUe9b
```



Note: The output from running this command should be **“Login successful for user ‘jim’”**.

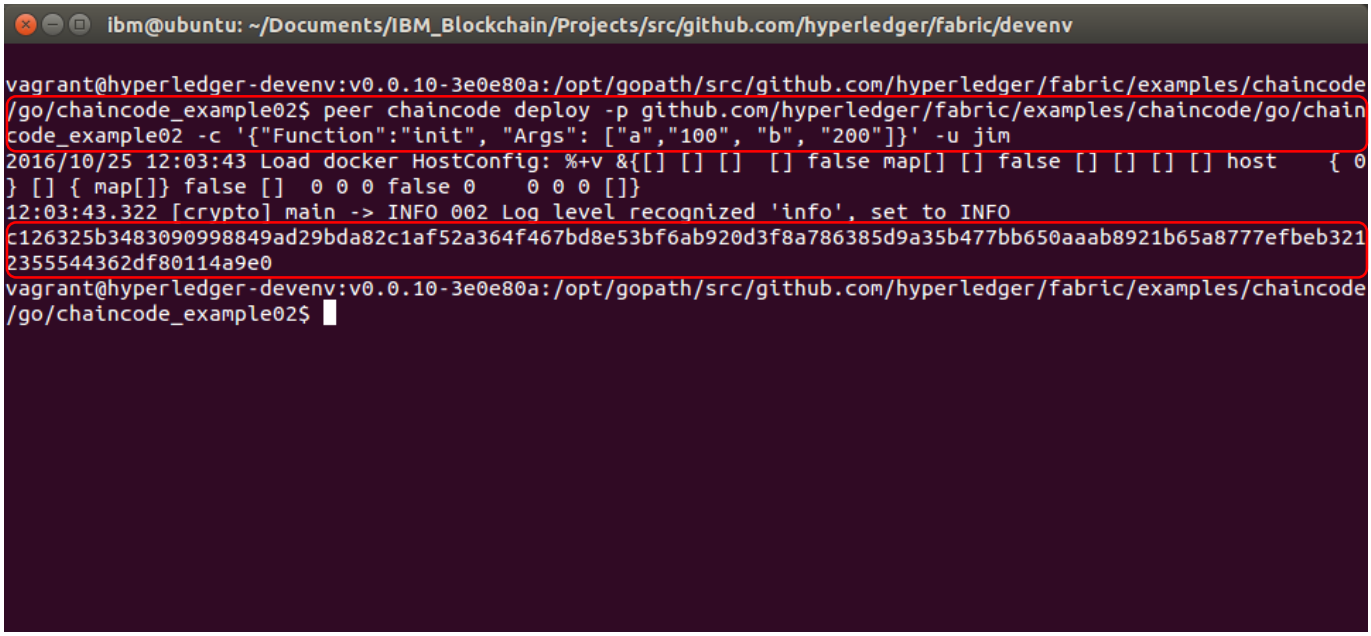
Now we can deploy the chaincode **example02** to the peer, using the same terminal where we have just logged in as **jim**.

Deploys can be done for chaincode which is located on the **`$GOPATH`** root directory, or in a public github repository. For the purposes of this lab we will continue to make use of the local **example02** sample. The full directory is not specified as it implies **`$GOPATH/src`** as the root.

__6. Enter the following **peer** command in the vagrant shell. The parameters passed in are used to call the **`Init()`** function reviewed earlier:

This command is also located in the file '**FYI – Cheatsheet Commands handout.pdf**' which can be found on the Ubuntu desktop in the VM. The command can be copied and pasted to the terminal window.

```
peer chaincode deploy -p
github.com/hyperledger/fabric/examples/chaincode/go/chaincode_example02
-c '{"Function":"init", "Args": ["a","100", "b", "200"]}' -u jim
```



Note: The output from running this command is the chaincode identifier:

`c126325b3483090998849ad29bda82c1af52a364f467bd8e53bf6ab920d3f8a786385d9a35b477bb650aaab8921b65a8777efbeb3212355544362df80114a9e0`

This will be different for each deploy, and uniquely identifies the chaincode deployed. Take a copy of the identifier from the deployment as it will be needed to invoke and query the chaincode.

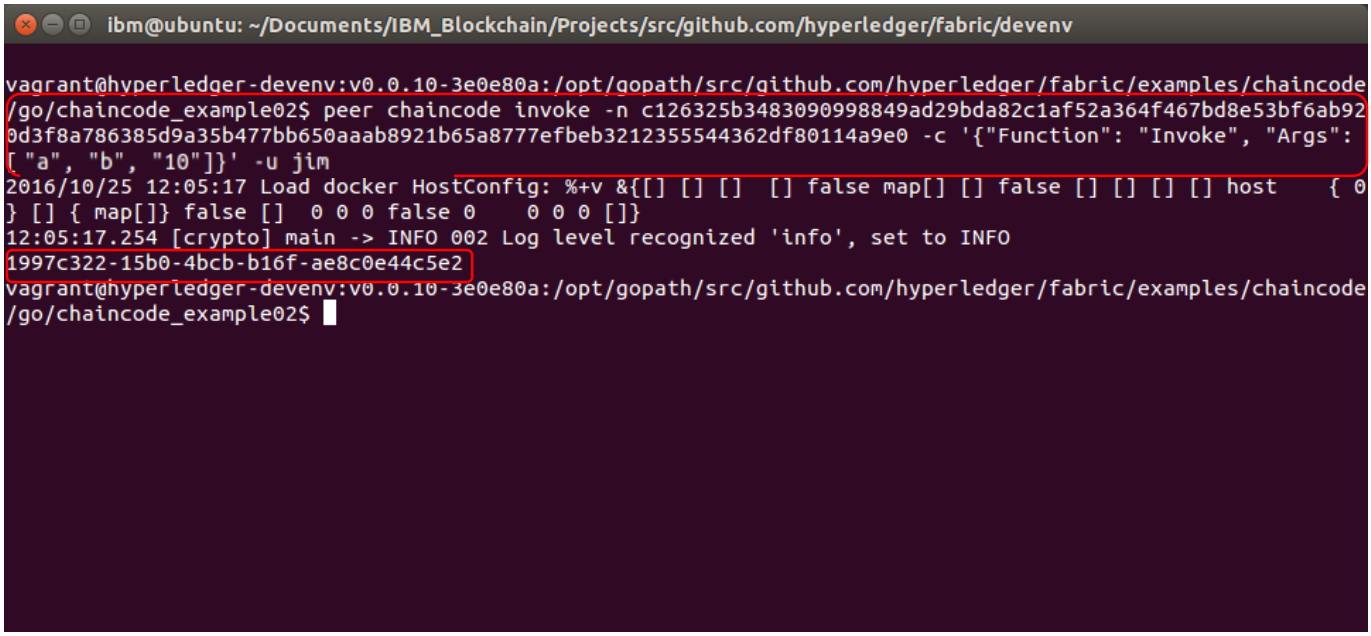
Section 4. Invoking the chaincode

We will now run the chaincode that has been deployed. This means submitting transactions to the validating peers that call the invoke method. This will add a new transaction to the blockchain.

__1. Enter the following **peer** command. The parameters passed in are used to call the **`Invoke()`** function we reviewed earlier. You will need to substitute the chaincode identifier with the one returned from the deployment in the previous step:

This command is also located in the file '**FYI – Cheatsheet Commands handout.pdf**' which can be found on the Ubuntu desktop in the VM. The command can be copied and pasted to the terminal window.

```
peer chaincode invoke -n <chaincode identifier from deployment> -c
'{"Function": "Invoke", "Args": ["a", "b", "10"]}' -u jim
```

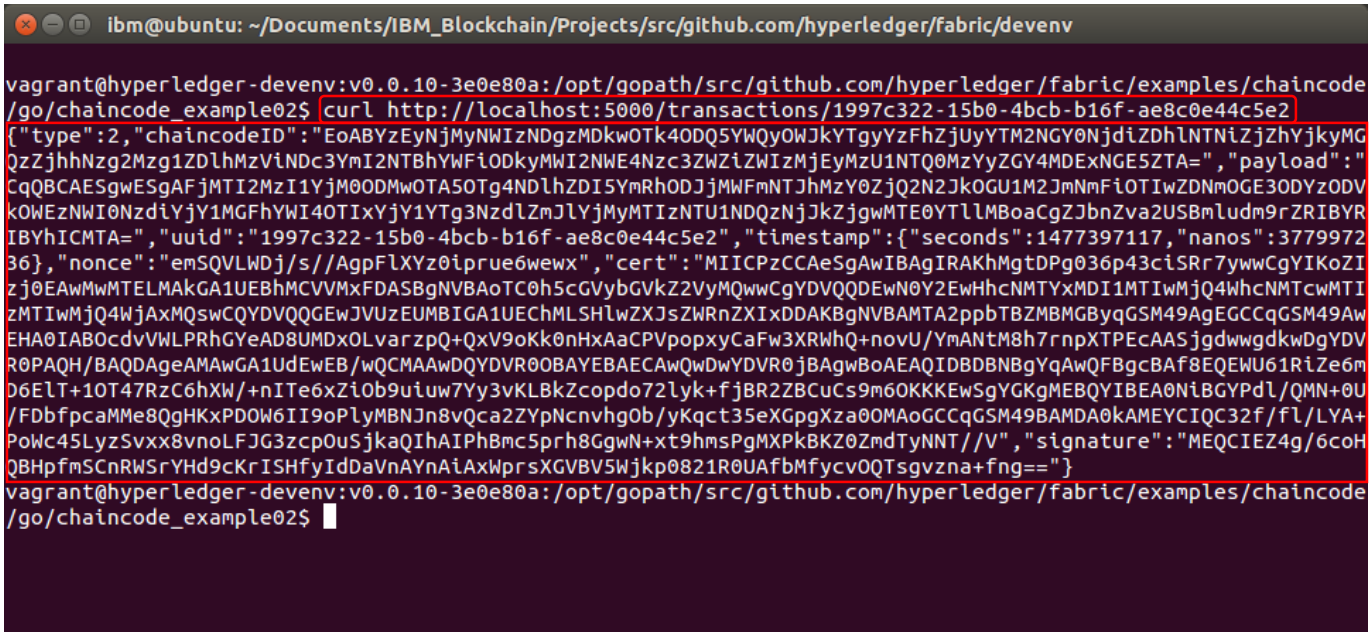


Note: The output from a successful **`Invoke()`** of the chaincode is a transaction UUID, for example:

```
1997c322-15b0-4bcb-b16f-ae8c0e44c5e2
```

__2. Review the transaction for UUID **`1997c322-15b0-4bcb-b16f-ae8c0e44c5e2`**, by querying the blockchain. To retrieve the transaction we must use the REST interface. Enter the following **curl** command, substituting the transaction UUID returned from the previous **`Invoke()`**.

```
curl http://localhost:5000/transactions/<transaction UUID>
```

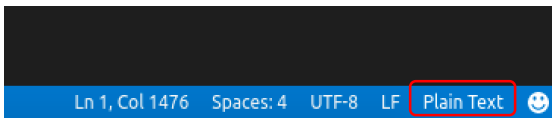


The output from this command is a JSON structure that includes all the information recorded on the blockchain for the transaction. Highlight and copy the JSON structure, so that we can review it in the Visual Studio Code editor:

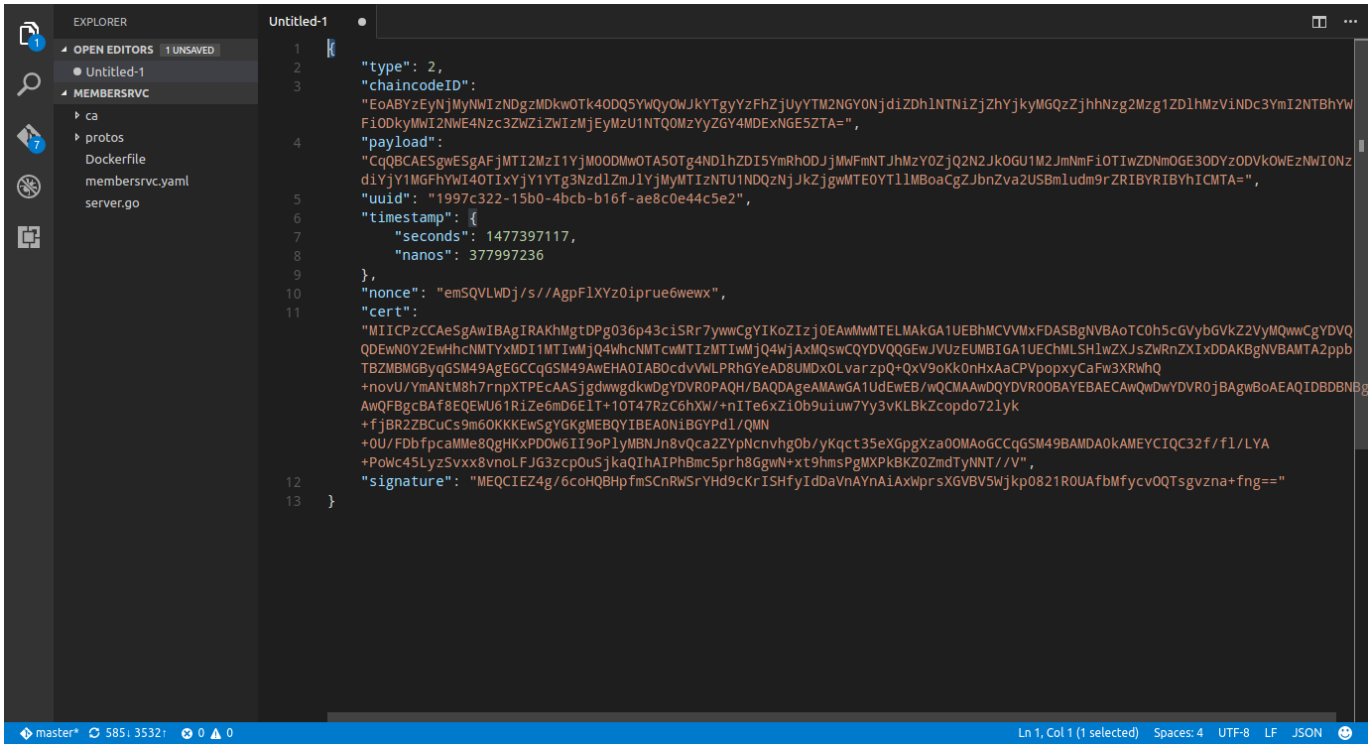
```
{"type":2,"chaincodeID":"EoABYzEyNjMyNWIzNDgzMDkwOTk4ODQ5YWQyOWJkYTgyYzFhZjUyYTM2NGY0NjdiZDhlNTNiZjZhYjkyM
GQzZjhhNzg2Mzg1ZDlhMzViNDc3YmI2NTBhYWFiODkyMWI2NWE4Nzc3ZWZiZWIzMjEyMzU1NTQ0MzYyZGY4MDExNGE5ZTA=","payload"
:"CqQBCAESgwESgAFjMTI2MzI1YjM0ODMwOTA5OTg4NDlhZDI5YmRhODJjMWFmNTJhMzY0ZjQ2N2JkOGU1M2JmNmFiOTIwZDNmOGE3ODYz
ODVkOWEzNWI0NzdiYjY1MGFhYWI4OTIxYjY1YTg3NzdlZmJlYjMyMTIzNTU1NDQzNjJkZjgwMTE0YTllMBoaCgZJbnZva2USBmludm9rZR
IBYRIBYhICMTA=","uuid":"1997c322-15b0-4bcb-b16f-
ae8c0e44c5e2","timestamp":{"seconds":1477397117,"nanos":377997236},"nonce":"emSQVLWDj/s//AgpFlXYz0iprue6we
wx","cert":"MIICPzCCAeSgAwIBAgIRAKhMgtDPg036p43ciSRr7ywwCgYIKoZIzj0EAwMwMTELMAkGA1UEBhMCVVMxFDASBgNVBAoTC0
h5cGVybGVkZ2VyMQwwCgYDVQQDEwN0Y2EwHhcNMTYxMDI1MTIwMjQ4WhcNMTcwMTIzMTIwMjQ4WjAxMQswCQYDVQQGEwJVUzEUMBIGA1UE
ChMLSHlwZXJsZWRnZXIxDDAKBgNVBAMTA2ppbTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABOcdvVWLPRhGYeAD8UMDxOLvarzpQ+QxV9
oKk0nHxAaCPVpopxyCaFw3XRWhQ+novU/YmANtM8h7rnpXTPEcAASjgdwwgdkwDgYDVR0PAQH/BAQDAgeAMAwGA1UdEwEB/wQCMAAwDQYD
VR0OBAYEBAECAwQwDwYDVR0jBAgwBoAEAQIDBDBNBgYqAwQFBgcBAf8EQEWU61RiZe6mD6ElT+1OT47RzC6hXW/+nITe6xZiOb9uiuw7Yy
3vKLBkZcopdo72lyk+fjBR2ZBCuCs9m6OKKKEwSgYGKgMEBQYIBEA0NiBGYPdl/QMN+0U/FDbfpcaMMe8QgHKxPDOW6II9oPlyMBNJn8vQ
ca2ZYpNcnvhgOb/yKqct35eXGpgXza0OMAoGCCqGSM49BAMDA0kAMEYCIQC32f/fl/LYA+PoWc45LyzSvxx8vnoLFJG3zcpOuSjkaQIhAI
PhBmc5prh8GgwN+xt9hmsPgMXPkBKZ0ZmdTyNNT//V","signature":"MEQCIEZ4g/6coHQBHpfmSCnRWSrYHd9cKrISHfyIdDaVnAYnA
iAxWprsXGVBV5Wjkp0821R0UAfbMfycvOQTsgvzna+fng=="}
```

__3. Switch back to the editor, and choose **File** > **New File** from the menu or press “**ctrl+n**” to create a new file. Paste the JSON output from the previous **curl** command into the new file.

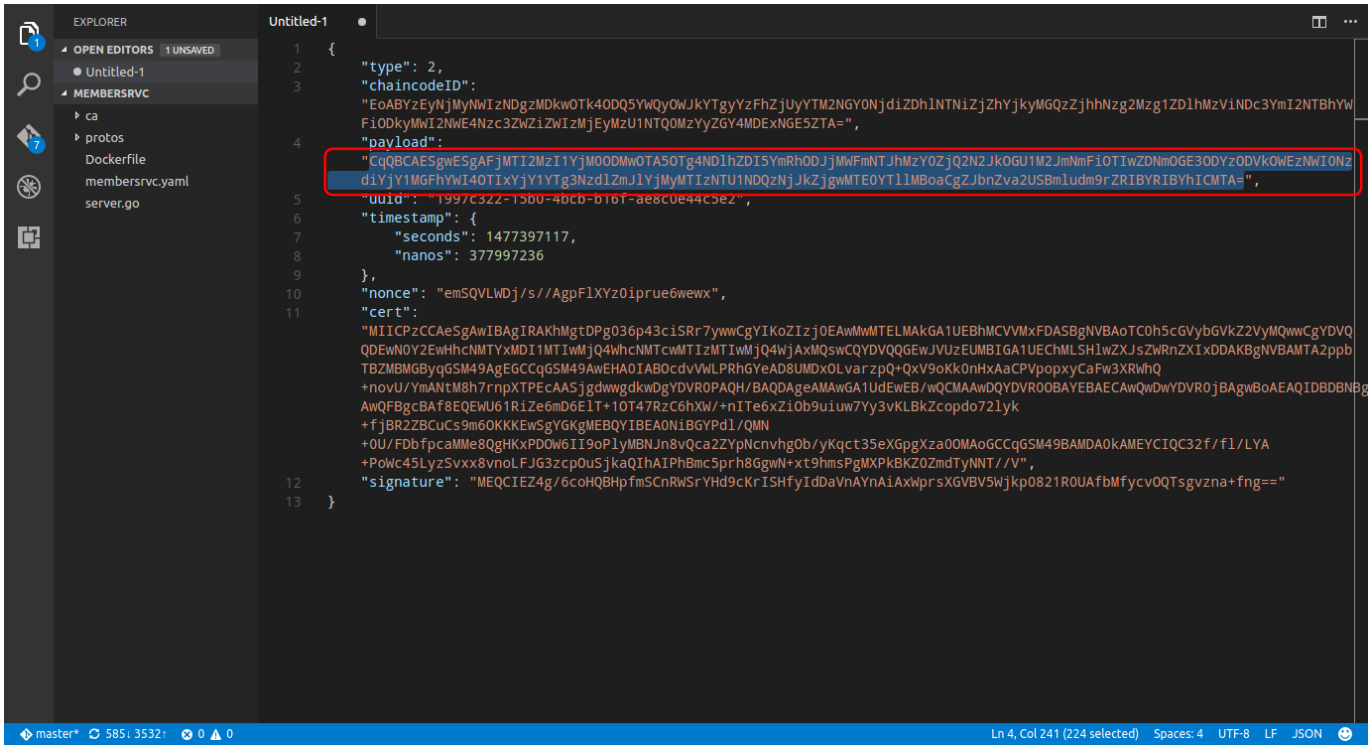
__4. In the bottom right of the editor window, click on “Plain Text” and from the list that appears choose “JSON” to set the editor format to show JSON data.



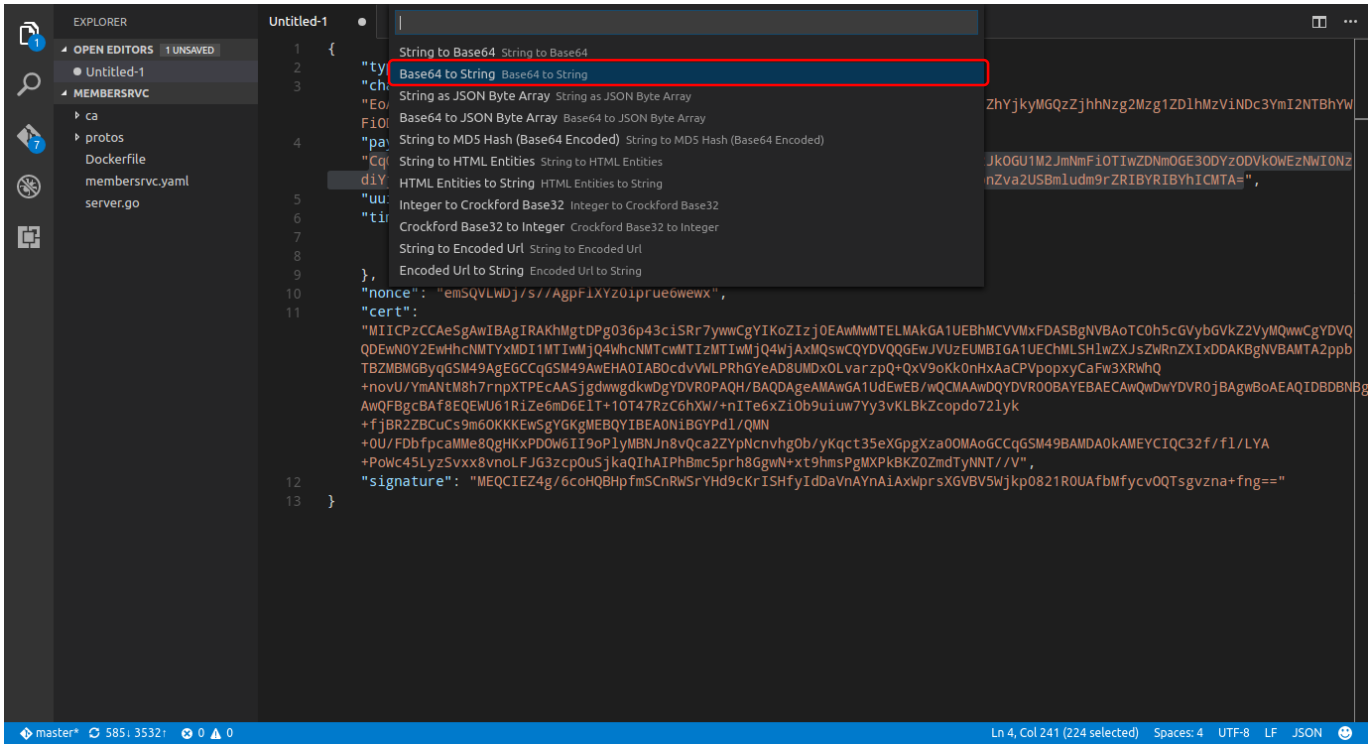
__5. You will see the text is colour coded, but not properly formatted yet. **Right-click** and choose ‘**Format Code**’. The text will now look similar to the screen shot below. You should also choose **View > Toggle Word Wrap** to wrap the text onto the screen.



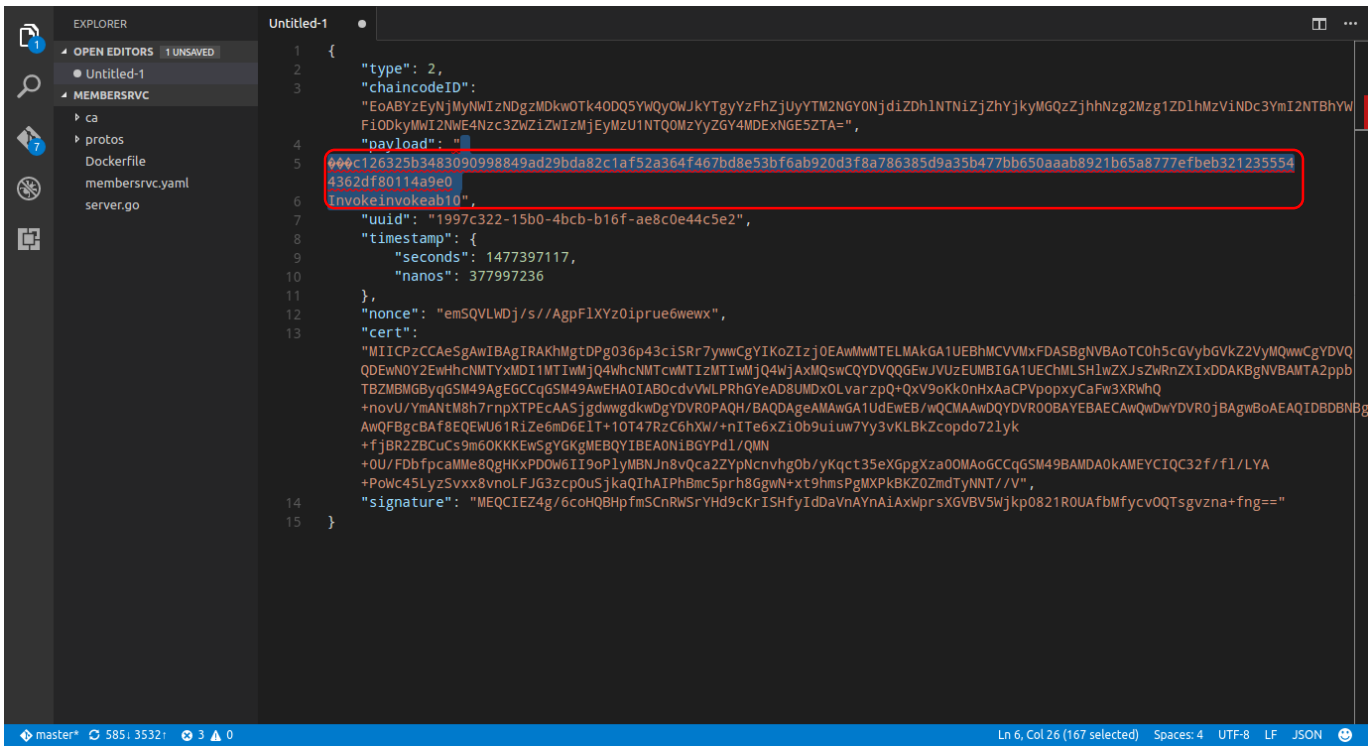
__6. Looking at the data you can see that much of it is not easily understood, because the data is also encoded using “base64” encoding. To see a little more of the contents we will decode the value of the “payload” field. First select all the data in the payload field inside the quotes, but do not select the quotes themselves.



__7. Now press “**ctrl+alt+c**” and the conversion menu will appear.



__8. From the conversion menu select the second option “**Base64 to String**” and the selected data will be decoded.



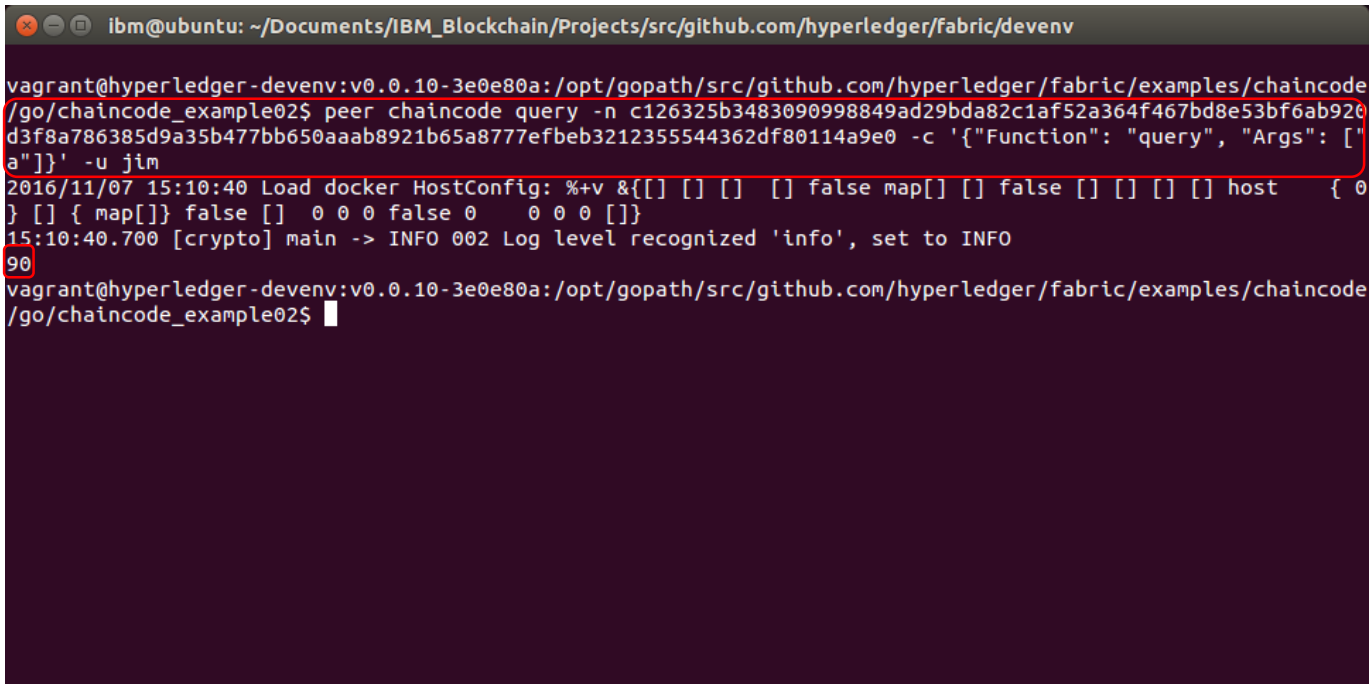
Section 5. Querying chaincode

Having run the chaincode successfully, it can now be queried.

__1. Enter the following **peer** command. The parameters passed in are used to call the **`Query()`** function we reviewed earlier. You will need to substitute the chaincode identifier with the one returned from the deployment in the previous step:

This command is also located in the file '**FYI – Cheatsheet Commands handout.pdf**' which can be found on the Ubuntu desktop in the VM. The command can be copied and pasted to the terminal window.

```
peer chaincode query -n <chaincode identifier from deployment> -c
'{"Function": "query", "Args": ["a"]}' -u jim
```



The output from running the query is the value **90** for the user **a**.

__2. Repeat the query for '**b**'.

__3. Rerun the **peer** invoke command, followed by the **peer** query command.

Congratulations on completing the Blockchain Unchained lab!

Appendix A. Notices

This information was developed for products and services offered in the U.S.A.
IBM may not offer the products, services, or features discussed in this document in other countries.

Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.
IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.
For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:
IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan
The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.
This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.
Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.
Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.
Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM

products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.
All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.
This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental. All references to fictitious companies or individuals are used for illustration purposes only.
COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Appendix B. Trademarks and copyrights

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	AIX	CICS	ClearCase	ClearQuest	Cloudscape
Cube Views	DB2	developerWorks	DRDA	IMS	IMS/ESA
Informix	Lotus	Lotus Workflow	MQSeries	OmniFind	
Rational	Redbooks	Red Brick	RequisitePro	System i	
System z	*Tivoli*	*WebSphere*	*Workplace*	*System p*	

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
ITIL is a registered trademark, and a registered community trademark of The Minister for the Cabinet Office, and is registered in the U.S. Patent and Trademark Office.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.
Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

NOTES

NOTES

The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. This information is based on current IBM product plans and strategy, which are subject to change by IBM without notice. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

Please Recycle