This is a guide of how to use jclouds implementation of ProfitBricks Cloud Compute Provider API directly and via ComputeService interface.

Requirements

JDK 7, jclouds 1.6.3+ (see jclouds install guide), jclouds ProfitBricks provider lib (sources are available here as a part of full clone of jclouds project), maven 3 in case you'll bootstrap from sources.

Restrictions

At the moment jclouds implementation of ProfitBricks API supports only Server and FIrewall Operations from the official API.

Ex1. Server Operations

First of all you need to create a ComputeServiceContext to specify the particular service you wish to manage and get a reference to ComputeService.

```
ComputeServiceContext ctx = ContextBuilder.newBuilder("profitbricks-c2")
        .credentials("user", "password")
        .buildView(ComputeServiceContext.class);

ComputeService cs = ctx.getComputeService();
```

Then you specify your future server's options (such as cpu, ram, etc.).

```
Template nodeCfg = cs.templateBuilder().build();

PBTemplateOptions pbOpts = nodeCfg.getOptions().as(PBTemplateOptions.class);
pbOpts.serverSpec(ServerCreationSpec.builder()
            .serverName("ServerName")
            .availabilityZone(AvailabilityZone.ZONE_1)
            .bootFromStorageId("sdb3abc85-b184-421d-8b74-7ead11becd35")
            .cores(1)
            .ram(1024)
            .internetAccess(true)
            .osType(OSType.LINUX)
            .build()
        ).addFirewallRuleSpec(FirewallRuleCreationSpec.builder()
            .protocol(IpProtocol.TCP)
            .fromPort(22)
            .toPort(22).build());
```

And calling ComputeService createNodesInGroup method you can create a new node.

```
Set<? extends NodeMetadata> nodes = cs.createNodesInGroup("grp", 1, nodeCfg);
NodeMetadata createdNode = Iterators.getOnlyElement(nodes.iterator());
```

Note that to be able to setup firewall rules in time of creating target node you need to set server's internet connection flag to true, otherwise there are no any network controllers will be created which is mandatory for firewall settings.

To list the nodes you have in all locations call the next method:

```
Set<? extend NodeMetadata> nodes = cs.listNodes();
for (ComputeMetadata node : nodes) {
  node.getId();
  node.getProviderId();
  node.getName();
  node.getLocation();
}
```

To get just the node's metadata call the next method:

```
cs.getNodeMetadata(nodeId)
```

To stop/start/restart the node perform next calls, respectively:

```
cs.suspendNode(nodeId);
cs.resumeNode(nodeId);
cs.rebootNode(nodeId);
```

To delete the node from cloud perform next call:

```
cs.destroyNode(nodeId);
```

And now, to update the node your way is to use a provider specific API directly:

```
PBApi pbApi = ctx.unwrapApi(PBApi.class);
pbApi.serversApi().updateServer(serverId, ServerUpdatingSpec.builder()
        .serverName("NewServerName")
        .cores(2)
        .ram(2048)
        .build()
);
```

But in that case you need to wait until provisioning will be completed (server's provision status must be AVAILABLE).

Ex2. Firewall Operations

As in the example above you have to get a Cloud Context. But in case of Firewall operations we forced to use a provider API directly (there is a SecurityGroupExtension API but it's in beta and most likely will be available in the future jclouds releases).

```
ComputeServiceContext ctx = ContextBuilder.newBuilder("profitbricks-c2")
        .credentials("user", "password")
        .buildView(ComputeServiceContext.class);

PBApi pbApi = ctx.unwrapApi(PBApi.class);
```

To list the nodes you have in all locations call the next method:

```
Set<Firewall> fws = pbApi.firewallApi().listFirewalls();
for (Firewall fw : fws) {
  fw.getId();
  fw.getNicId();
  fw.isActive();
  fw.getRules();
}
```

To get just the firewall's detailed metadata call the next method:

```
Firewall fw = pbApi.firewallApi().getFirewall(fwId);
```

To activate/deactivate/remove the firewall perform next calls, respectively:

```
pbApi.firewallApi().activateFirewall(fwId);
pbApi.firewallApi().deactivateFirewall(fwId);
pbApi.firewallApi().deleteFirewall(fwId);
```

To add new firewall rule to the target network interface controller perform next steps:

```
FirewallRuleCreationSpec ruleSpec = FirewallRuleCreationSpec.builder()
        .sourceIp("172.168.38.3")
        .protocol(IpProtocol.TCP)
        .fromPort(1)
        .toPort(400)
        .build();

pbApi.firewallApi().addFirewallRule(nicId, ruleSpec);
```

To remove a certain rule from the firewall you should know just a rule ID and perform next call:

```
pbApi.firewallApi().removeFirewallRule(ruleId);
```

But again, note that case you need to check firewall's provision state before executing any updating command. Implementation of that check is up to consumer of jclouds ProfitBricks provider library and in the most trivial case can look like the code below:

```
  void blockUntilFirewallAvailableState(String firewallId) throws InterruptedException {
    final int RETRY_NUM = 200;

    for (int i = 0; i < RETRY_NUM; i++) {
      Firewall firewall = providerApi.firewallApi().getFirewall(firewallId);
      if (firewall.getProvisioningState() == ProvisioningState.AVAILABLE)
        return;
      else
        Thread.sleep((long) 5*1000); // wait for 5 sec
    }

    throw new TimeoutException();
  }
```