cloudstack

open source cloud computing

Developing API Plug-ins for CloudStack*

* Specifically Using Version 4.5

SOLIDFIRE

Mike Tutkowski (@mtutkowski on Twitter)

- CloudStack Software Engineer
- Member of CloudStack Project Management Committee (PMC)
- Focused on CloudStack's storage component

NetApp SolidFire (http://www.solidfire.com/)

- Based out of Boulder, CO, USA
- Develop a scale-out storage technology (using industry-standard hardware)
- Built from the ground up to support guaranteed Quality of Service (QoS) on a per-volume (logical unit) basis (min, max, and burst IOPS per volume)
- All-Flash Array
- Leverage compression, de-duplication, and thin provisioning (all inline) on a 4-KB block boundary across the entire cluster to drive down cost/GB to be on par with traditional disk-based systems
- Rest-like API to enable automation of all aspects of the SAN

Why might I want to develop an API Plug-in for CloudStack?

SOLIDFIRE

- There is a feature in a third-party product that you would like to give your CloudStack end users and/or admins access to without the need to change core CloudStack logic (ex. A SAN that supports virtual networks).

- Wrapping this functionality in a CloudStack plug-in decouples your development process from that of the CloudStack community (you can release your plug-in against particular versions of CloudStack on your own schedule).

What's our approach here?

SOLIDFIRE

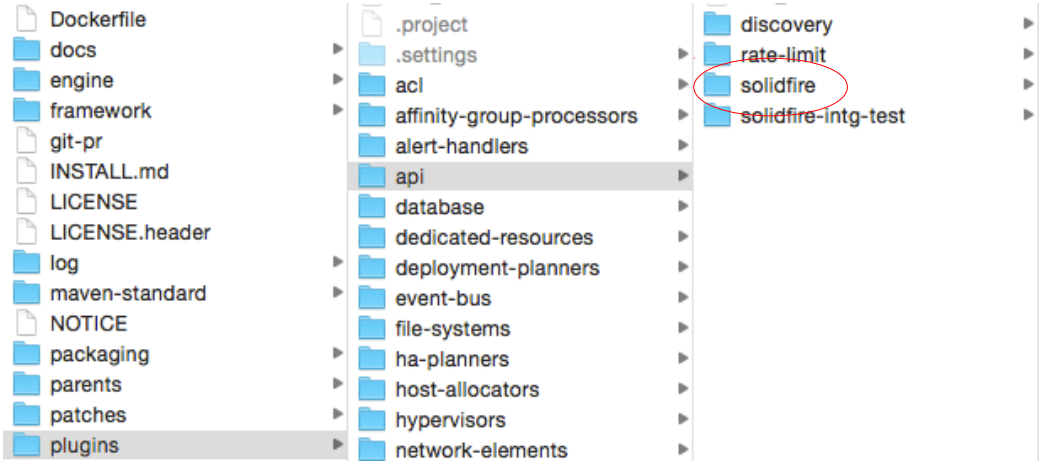
We'll construct a basic API plug-in step by step using an existing API plug-in as a template. This plug-in will enable us to extend CloudStack's standard API with new commands.

- These are invoked in the same manner as any standard CloudStack API command.
- The client does not know if the API command is standard or an extension.
- The client can discover these extended API commands just like standard API commands (using the listApis API command).
- Multiple API plug-ins can all run at the same time providing many extended API commands.

Creating an API plug-in for CloudStack

SOLIDFIRE

- Relative to CloudStack's root folder, let's leverage an existing Maven project in plugins/api as a template for yours.
- We will use a SolidFire project as a template for a new project called “abc123”.



Creating an API plug-in for CloudStack

SOLIDFIRE

- Relative to the root abc123 folder (created on the previous slide by copying and pasting the “solidfire” folder and naming the new folder “abc123”), update the pom.xml file with applicable information.

```
<modelVersion>4.0.0</modelVersion>
<artifactId>cloud-plugin-api-solidfire</artifactId>
<name>SolidFire API Plugin</name>
<parent>
  <groupId>org.apache.cloudstack</groupId>
  <artifactId>cloudstack-plugins</artifactId>
  <version>4.5.2-SNAPSHOT</version>
  <relativePath>../../pom.xml</relativePath>
</parent>
<dependencies>
    <dependency>
        <groupId>org.apache.cloudstack</groupId>
        <artifactId>cloud-plugin-storage-volume-solidfire</artifactId>
        <version>${project.version}</version>
    </dependency>
</dependencies>
```

Creating an API plug-in for CloudStack

SOLIDFIRE

- Relative to the root abc123 folder, find all folders and files that have the text “solidfire” in them and change them to reference the text “abc123” (ex. spring-solidfire-context.xml → spring-abc123-context.xml).

- Relative to the root abc123 folder, locate the resources/META-INF/cloudstack/abc123/module.properties file.

- In this file, update the text “solidfire” to the text “abc123”.

```
name=solidfire
parent=api
```

Creating an API plug-in for CloudStack

SOLIDFIRE

- Relative to the root abc123 folder, locate the resources/META-INF/cloudstack/abc123/spring-abc123-context.xml file.
- In this file, only keep three of the <bean/> lines.
- Update the <bean/> lines such as below (the id attribute is not currently used, but each class attribute must reference a class we are to make later).

```
<bean id="abc123Util" class="org.apache.cloudstack.util.abc123.Abc123Util" />
<bean id="abc123ManagerImpl" class="org.apache.cloudstack.abc123.Abc123ManagerImpl" />
<bean id="apiAbc123ServiceImpl" class="org.apache.cloudstack.api.abc123.ApiAbc123ServiceImpl" />
```

Creating an API plug-in for CloudStack

SOLIDFIRE

- Relative to CloudStack's root folder, locate the plugins/pom.xml file.
- Add the following line into the <modules> section.

```
<module>api/abc123</module>
```

Creating an API plug-in for CloudStack

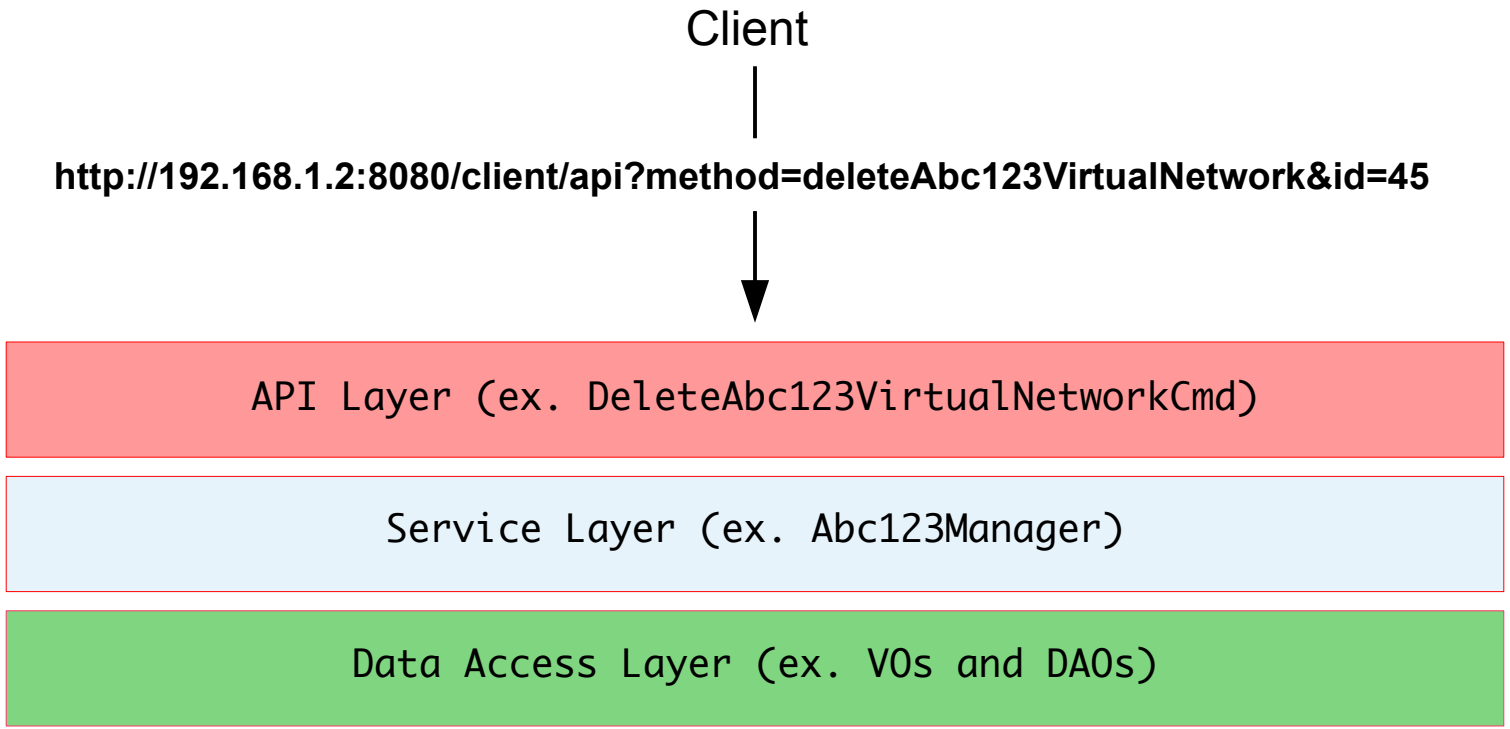
SOLIDFIRE

- Relative to CloudStack's root folder, update the <dependencies> section of the client/pom.xml file to reference your new project.

```
<dependency>
    <groupId>org.apache.cloudstack</groupId>
    <artifactId>cloud-plugin-api-abc123</artifactId>
    <version>${project.version}</version>
</dependency>
```

Creating an API plug-in for CloudStack

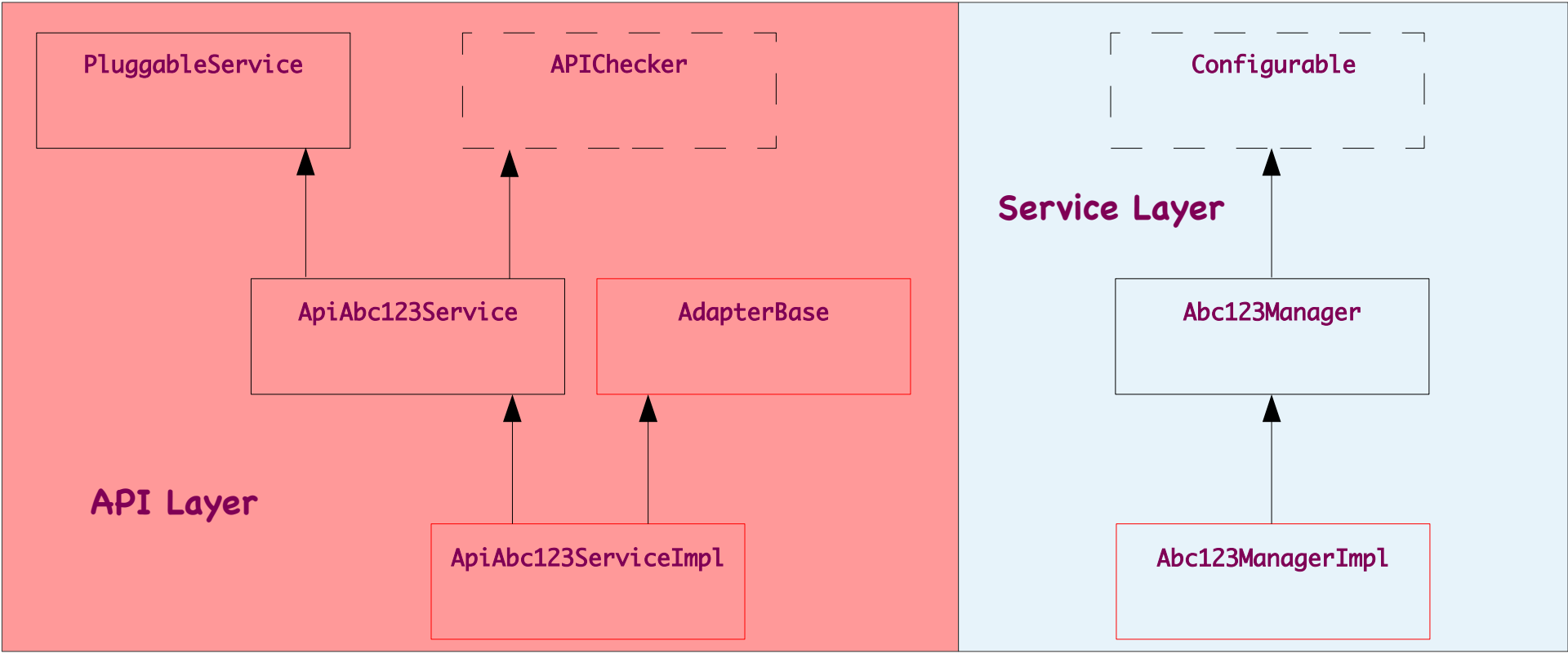
SOLIDFIRE



Note: A layer only knows about the layer right below it. It does not know about any layers above it.

Creating an API plug-in for CloudStack

SOLIDFIRE



Note: Black box = Interface; Red box = Class; Dotted box = Optional Interface

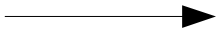
Creating an API plug-in for CloudStack

SOLIDFIRE

Service Layer

```
package org.apache.cloudstack.abc123;

import org.apache.cloudstack.framework.config.Configurable;

public interface Abc123Manager extends Configurable {
```

```
public interface Configurable {

    String getConfigComponentName();

    ConfigKey<?>[] getConfigKeys();
}
```



```
// example way to implement this
return Abc123ManagerImpl.class.getSimpleName();
```

Creating an API plug-in for CloudStack

SOLIDFIRE

Service Layer

```
public class Abc123ManagerImpl implements Abc123Manager {

    private static final ConfigKey<Long> s_TotalAccountCapacity =
            new ConfigKey<>(
                    "Advanced",
                    Long.class,
                    "abc123.total.capacity",
                    "0",
                    "Total capacity the account can draw from any and all clusters (in GBs)",
                    true, ConfigKey.Scope.Account);

    @Override
    public ConfigKey<?>[] getConfigKeys() {
        return new ConfigKey<?>[] { s_TotalAccountCapacity };
    }
```

Creating an API plug-in for CloudStack

SOLIDFIRE

Service Layer

```
public interface Abc123Manager extends Configurable {

    public List<Abc123VirtualNetwork> listAbc123VirtualNetworks(Long id);

    public Abc123VirtualNetwork deleteAbc123VirtualNetwork(long id);

}
```

If number and required, I use a primitive.

If number and NOT required, I use a number wrapper and check for null to see if it was not provided.

Creating an API plug-in for CloudStack

SOLIDFIRE

Service Layer

```
public class Abc123ManagerImpl implements Abc123Manager {
  @Override
  public Abc123VirtualNetwork deleteAbc123VirtualNetwork(long id) {
      verifyRootAdmin();

      Abc123VirtualNetworkVO virtualNetwork = getAbc123VirtualNetworkVO(id);

      List<Abc123VolumeVO> volumes = _abc123VolumeDao.findByAbc123VirtualNetworkId(virtualNetwork.getId());

      if (volumes != null && volumes.size() > 0) {
          throw new CloudRuntimeException("Unable to delete a virtual network that has one or more volumes");
      }

      if (!_abc123VirtualNetworkDao.remove(id)) {
          throw new CloudRuntimeException("Unable to remove the following virtual network: " + id);
      }

      Abc123ClusterVO cluster = getAbc123ClusterVO(virtualNetwork.getAbc123ClusterId());

      Abc123Connection conn = new Abc123Connection(cluster.getIp(), cluster.getUsername(), cluster.getPassword());

      conn.deleteVirtualNetwork(virtualNetwork.getAbc123Id());

      return virtualNetwork;
  }
```

Return interface type (ex. Not Abc123VirtualNetworkVO) so caller is more abstracted away from where this data lives (in case its location is changed in the future and the underlying type needs to change, too).

Creating an API plug-in for CloudStack

SOLIDFIRE

API Layer

```
package org.apache.cloudstack.abc123;

import com.cloud.utils.component.PluggableService;

import org.apache.cloudstack.acl.APIChecker;

public interface ApiAbc123Service extends PluggableService, APIChecker {
```

```
public interface PluggableService {
    List<Class<?>> getCommands();
}
```

```
// optional: only required if we have special needs with regards to checking API permissions
public interface APIChecker extends Adapter {
    boolean checkAccess(User user, String apiCommandName) throws PermissionDeniedException;
}
```

Creating an API plug-in for CloudStack

SOLIDFIRE

API Layer

```
public class ApiAbc123ServiceImpl extends AdapterBase implements ApiAbc123Service {

    @Override
    public List<Class<?>> getCommands() {
        List<Class<?>> cmdList = new ArrayList<Class<?>>();

        cmdList.add(ListAbc123VirtualNetworksCmd.class);

        cmdList.add(DeleteAbc123VirtualNetworkCmd.class);

        return cmdList;
    }
```

Creating an API plug-in for CloudStack

SOLIDFIRE

API Layer

- Relative to CloudStack's root folder, locate the client/tomcatconf/commands.properties.in file.

- Add the following lines into the file (the meaning of the numbers is provided at the top of the file):

```
#### API Abc123 Service Commands
deleteAbc123VirtualNetwork=1
listAbc123VirtualNetworks=15
```

- If an API command is not listed in this file, the StaticRoleBasedAPIAccessChecker (which implements the APIChecker interface) will cause CloudStack to not “see” the command.

Creating an API plug-in for CloudStack

SOLIDFIRE

API Layer

```
public class ApiAbc123ServiceImpl extends AdapterBase implements ApiAbc123Service {

    @Override
    public boolean checkAccess(User user, String apiCommandName) throws PermissionDeniedException {
        if (_accountMgr.isRootAdmin(user.getAccountId())) {
            return true;
        }

        if ("listAbc123VirtualNetworks".equals(apiCommandName)) {
            return true;
        }

        throw new PermissionDeniedException("User " + user.getFirstname() + " " +
            user.getLastname() + " cannot access the following command: " + apiCommandName);
    }
```

Creating an API plug-in for CloudStack

SOLIDFIRE

API Layer

```
@APICommand(name = "deleteAbc123VirtualNetwork",
    responseObject = ApiAbc123VirtualNetworkResponse.class,
    description = "Delete Abc123 Virtual Network",
    requestHasSensitiveInfo = false, responseHasSensitiveInfo = false)
public class DeleteAbc123VirtualNetworkCmd extends BaseCmd {
    private static final Logger s_logger =
        Logger.getLogger(DeleteAbc123VirtualNetworkCmd.class.getName());
    private static final String s_name = "deleteabc123virtualnetworkresponse";

    @Parameter(name = ApiConstants.ID, type = CommandType.UUID,
        entityType = ApiAbc123VirtualNetworkResponse.class,
        description = ApiHelper.VIRTUAL_NETWORK_ID_DESC, required = true)
    private long _id;

    @Inject private Abc123Util _abc123Util;
    @Inject private Abc123Manager _abc123Manager;
```

Creating an API plug-in for CloudStack

SOLIDFIRE

API Layer

```
@Override
public String getCommandName() {
    return s_name;
}

@Override
public long getEntityOwnerId() {
    return _accountId;
}
```

Creating an API plug-in for CloudStack

SOLIDFIRE

API Layer

```
@Override
public void execute() {
    try {
        s_logger.info(DeleteAbc123VirtualNetworkCmd.class.getName() + ".execute invoked");

        Abc123VirtualNetwork virtualNetwork = _apiAbc123Manager.deleteAbc123VirtualNetwork(_id);

        ApiAbc123VirtualNetworkResponse response =
            _abc123Util.getApiAbc123VirtualNetworkResponse(virtualNetwork, ResponseView.Full);

        response.setResponseName(getCommandName());
        response.setObjectName("apideleteabc123virtualnetwork");

        setResponseObject(response);
    }
    catch (Throwable t) {
        s_logger.error(t.getMessage());

        throw new ServerApiException(ApiErrorCode.INTERNAL_ERROR, t.getMessage());
    }
}
```

Creating an API plug-in for CloudStack

SOLIDFIRE

API Layer

```
@EntityReference(value = Abc123VirtualNetwork.class)
public class ApiAbc123VirtualNetworkResponse extends BaseResponse {
    @SerializedName("id")
    @Param(description = "CloudStack ID")
    private long _id;

    @SerializedName("uuid")
    @Param(description = "CloudStack UUID")
    private String _uuid;

    @SerializedName("name")
    @Param(description = ApiHelper.VIRTUAL_NETWORK_NAME_DESC)
    private String _name;

    @SerializedName("size")
    @Param(description = ApiHelper.SIZE_DESC)
    private int _size;
```

Creating an API plug-in for CloudStack

SOLIDFIRE

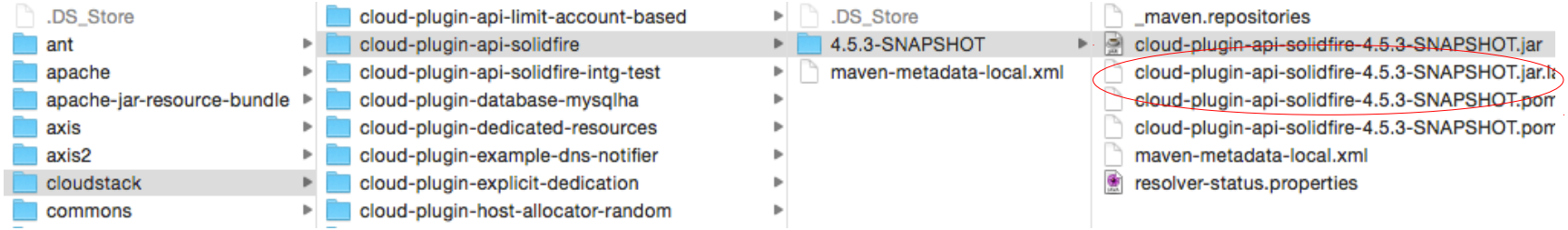
Translate Service-Layer Object to API-Layer Object

```
public ApiAbc123VirtualNetworkResponse getApiAbc123VirtualNetworkResponse(Abc123VirtualNetwork abc123VirtualNetwork, ResponseView responseView) {
    ApiAbc123VirtualNetworkResponse abc123Response = new ApiAbc123VirtualNetworkResponse();

    abc123Response.setId(abc123VirtualNetwork.getId());
    abc123Response.setUuid(abc123VirtualNetwork.getUuid());
    abc123Response.setAccountId(abc123VirtualNetwork.getAccountId());

    Account account = _accountDao.findById(abc123VirtualNetwork.getAccountId());

    abc123Response.setAccountUuid(account.getUuid());
    abc123Response.setAccountName(account.getAccountName());

    Abc123Cluster abc123Cluster = _abc123ClusterDao.findById(abc123VirtualNetwork.getAbc123ClusterId());

    abc123Response.setZoneId(abc123Cluster.getZoneId());

    DataCenterVO dataCenterVO = _zoneDao.findById(abc123Cluster.getZoneId());

    abc123Response.setZoneUuid(dataCenterVO.getUuid());
    abc123Response.setZoneName(dataCenterVO.getName());

    if (ResponseView.Full.equals(responseView)) {
        abc123Response.setClusterName(abc123Cluster.getName());
    }

    abc123Response.setObjectName("abc123virtualnetwork");

    return abc123Response;
}
```

Creating an API plug-in for CloudStack

SOLIDFIRE

Deploying your API Plug-in



Shut down the CloudStack Management Server.

Place cloud-plugin-api-abc123-4.5.3-SNAPSHOT.jar in /usr/share/cloudstack-management/webapps/client/WEB-INF/classes.

Updated commands.properties should be placed here: /usr/share/cloudstack-management/webapps/client/WEBINF/classes

Restart the CloudStack Management Server.