KNOX-177: Simplify service deployment contributor implementation

This design is intended to address the following issues:

KNOX-177@jira: Simplify service deployment contributor implementation

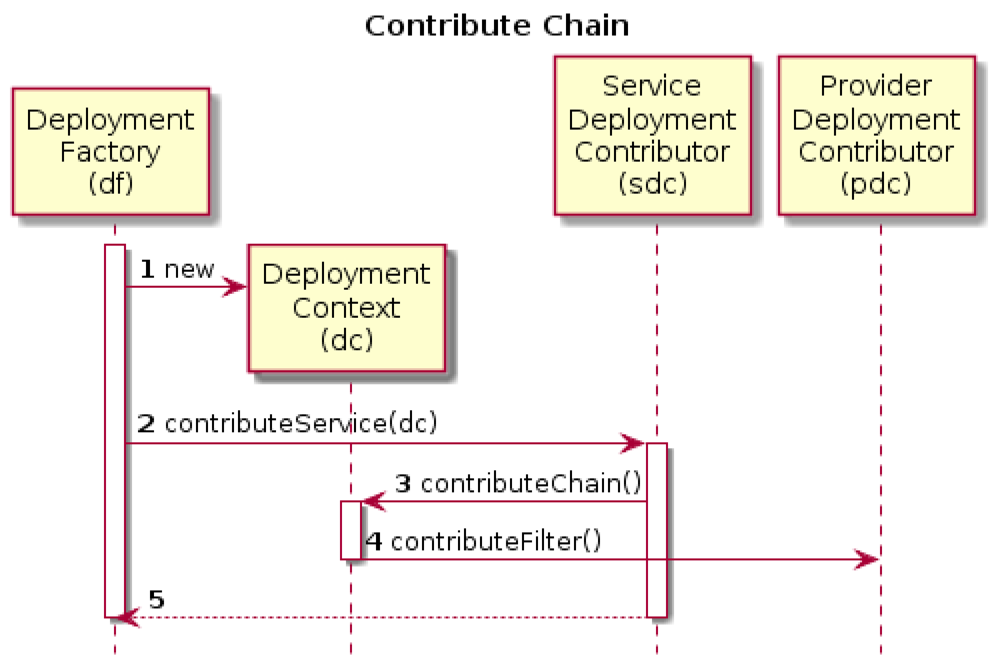
This issue stems from what is currently expected of a `ServiceDeploymentContributor` in its `contributeService` method.
Basically each service deployment contributor is expected to build its own filter chain.
This is currently done by making calls to `Deploymentcontext.contributeFilter`.
While this provides a great deal of flexibility for each service to define a custom chain we have found that this isn't commonly necessary.
Furthermore it makes if very difficult if not impossible to introduce new provider/filters without impacting all services.
This design will provide an abstraction to the service deployment contributors that can create either a default or specifically configured chain of providers /filters.

The goal is to support a pattern in service deployment contributors that looks like this:

ServiceDeploymentContributor.contributeService

```
  public void contributeService( DeploymentContext context, Service service ) throws Exception {
    List<ServiceParamDescriptor> params = null; // Default if null, otherwise map of per provider role map of
name/value pairs.
    ResourceDescriptor resource = context.addResource()
    resource.role( "WEBHDFS" );
    resource.pattern( "webhdfs/v1/**?**" );
    context.contributeChain( service, resource, params );
  }
```

The over all context into which the method above fits into the deployment infrastructure is shown in the sequence diagram below.



The method below shows a more complete example of a contributeService method as it might be implemented for WebHDFS.

WebHdfsDeploymentContributor.contributeService

```
  public void contributeService( DeploymentContext context, Service service ) throws Exception {
    UrlRewriteRulesDescriptor serviceRules = loadRulesFromClassPath();
    UrlRewriteRulesDescriptor clusterRules = context.getDescriptor( "rewrite" );
    clusterRules.addRules( serviceRules );

    ResourceDescriptor resource;
    List<ServiceParamDescriptor> params;

    resource = context.getGatewayDescriptor().addResource();
    resource.role( "WEBHDFS" );
    resource.pattern( "webhdfs/v1/?**" );
    resource.pattern( "webhdfs/v1/**?**" );
    params = new ArrayList<ServiceParamDescriptor>();
    params.add( resource.createParam().role( "rewrite", "request.url", "/webhdfs/namenode/inbound/path" );
    params.add( resource.createParam().role( "rewrite", "response.headers", "/webhdfs/namenode/outbound
/headers" );
    context.contributeChain( service, resource, params );

    resource = context.getGatewayDescriptor().addResource();
    resource.role( "WEBHDFS" );
    resource.pattern( "webhdfs/data/v1/**?**" );
    params = new ArrayList<ServiceParamDescriptor>();
    params.add( resource.createParam().role( "rewrite", "request.url", "/webhdfs/datanode/inbound/path" );
    context.contributeChain( service, resource, params );
  }
```

This is a sketch of how topology files would need to be extended to support the external chain definitions.
See table below for details on the new elements introduced.
Note: I don't really like the names for provider-ref and chain-ref but I can't come up with anything better.

Sample Topology Descriptor

```
<topology>

  <gateway>

    <provider>
      <role>...</role>
      <name>...</name>
      <param><name>...</name><value>...</value></param>
    </provider>

    <chain>
      <name>...</name>
      <provider-ref>
        <role>...</role>
        <name>...</name>
        <param><name>...</name><value>...</value></param>
      </provider-ref>
      <provider-ref>...</provider-ref>
    </chain>
    <chain>...</chain>

  </gateway>

  <service>
    <role>...</role>
    <url>...</url>
    <chain-ref>
      <name>...</name>
      <param><role></role><name></name><value></value></param>
    </chain-ref>
    <param><name></name><value></value></param>
  </service>

</topology>
```

Details on the new elements within the topology are described below.

Path	Description
topology/gateway/chain	This defines a new chain structure and configuration for use by services. There will be a "built-in" chain named "default".
topology/gateway/chain/name	Specifies the name of the chain so that it can be referenced by services.
topology/gateway/chain/provider-ref	References a configured or default provider. May repeat.
topology/gateway/chain/provider-ref/role	A required role of a provider to be included in the chain.
topology/gateway/chain/provider-ref/name	An optional name of a specific provider for the given role.
topology/gateway/chain/provider-ref/param	Optional config parameters to augment the provider's configuration.
topology/service/chain-ref	Selects a specific chain to use for the service. May repeat.
topology/service/chain-ref/name	Specifies the name of the chain to use for the service. Default is "default"
topology/service/chain-ref/param	Optional parameters to augment the chain and provider configuration.
topology/service/chain-ref/param/role	A role name to disambiguate which provider the param is intended.
topology/service/param	Configuration parameters used by the service. May repeat.

This shows the new method `contributeChain()` that would be added to the DeploymentContext interface.
The existing contributeFilter method would be deprecated.
This is actually a point worth further discussion.
Is there a use case where a service might want to define a chain this way?

DeploymentContext

```
public interface DeploymentContext {
  ...
  @Deprecated
  void contributeFilter(
      Service service,
      ResourceDescriptor resource,
      String role,
      String name,
      List<FilterParamDescriptor> params );

  void contributeChain(
      Service service,
      ResourceDescriptor resource,
      List<ServiceParamDescriptor> params );
  ...
}
```

This shows the relevant portions of the existing, unmodified ServiceDeploymentContributor interface.

ServiceDeploymentContributor

```
public interface ServiceDeploymentContributor {
  ...
  // Called per service based on the service's role.
  void contributeService( DeploymentContext context, Service service ) throws Exception;
  ...
}
```

This shows the relevant portions of the existing, unmodified ProviderDeploymentContributor interface.

ProviderDeploymentContributor

```
public interface ProviderDeploymentContributor {
  ...
  // This will be called indirectly by a ServiceDeploymentContributor when it asks
  // for a chain to be contributed via DeploymentContext.contributeChain.
  void contributeFilter(
      DeploymentContext context,
      Provider provider,
      Service service,
      ResourceDescriptor resource,
      List<FilterParamDescriptor> params );
  ...
}
```