# Setup core components

1. Solution Explorer -> Add new project -> Console App (.Net Framework) -> IronMacbeth.<Name>
2. Solution Explorer -> Add new project -> Class Library (.Net Framework) -> IronMacbeth.<Name>.Contract
3. In contract project add I<Name>Service interface. Mark interface with [ServiceContract]. Add reference to System.ServiceModel.dll as Visual Studio (VS) suggests. Do not forget to mark methods with [OperationContract]. Put all classes that are part of contract in contract project, but nothing more.
4. In service project (IronMacbeth.<Name>) add <Name>Service class and implement interface from previous step (Add reference to contract project as VS suggests).
5. In service project go to app.config and add wcf configuration template from [docs.microsoft.com](https://docs.microsoft.com/en-us/dotnet/framework/wcf/samples/simplified-configuration-for-wcf-services). Add wsHttpBinding configuration template from [docs.microsoft.com](https://docs.microsoft.com/en-us/dotnet/framework/wcf/how-to-specify-a-service-binding-in-configuration).
6. In service project go to program.cs and add code for starting wcf service ([docs.microsoft.com](https://docs.microsoft.com/en-us/dotnet/framework/wcf/how-to-host-a-wcf-service-in-a-managed-application) section “Create a self-hosted service”, item 6.). Omit configuration, no need to configure from C#, in our case configuration is App.config. Add reference to System.ServiceModel.dll as VS suggests.
7. In service project go to app.config update configuration -> system.serviceModel -> services -> service (late referred to as wcf service configuration in app.config) to match your case:
   1. Set name = IronMacbeth.<Name>.<Name>Service (namespace + class name)
   2. Set behaviorConfiguration = IronMacbeth.<Name>.Behavior (or whatever, up to you). Note: Also, update behavior name in behaviors accordingly.
   3. Make sure there is only one endpoint (that’s all we need)
   4. Set address = <Name> (can be whatever).
   5. Set binding = wsHttpBinding
   6. Set contract = IronMacbeth.<Name>.Contract.I<Name>Service (namespace + interface name)
   7. Add bindingConfiguration = IronMacbeth.<Name>.Binding (or whatever). Note: update bindings config section accordingly.
   8. Add host section ([docs.microsoft.com](https://docs.microsoft.com/en-us/dotnet/framework/configure-apps/file-schema/wcf/host)). Ignore timeouts section. Set base address to [http://localhost:<Port](http://localhost:%3cPort)>. Note: port should not be already in use by other service.
8. Set service project as startup project:  
   
9. Launch project. There should not be any errors.
10. Create a new Console app (it’s just to test that everything works)
    1. Here are program.cs and app.config.  
       
    2. Set correct values in Program.cs and App.config.
    3. In VS set multiple startup projects (Test console app & service)
    4. Launch. Client to server communication should be working. Try sending arbitrary data: strings, ints, DateTimes… whatever.

# Setup security

1. Go to test app. Enable certificate security [docs.microsoft.com](https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/message-security-with-a-certificate-client) (Client -> Configuration)
   1. Add behaviors (here is a search criteria for current certificate: findValue="77f7f5d4aa2d038c4a1324df82f1bbe4" storeLocation="LocalMachine" storeName="My" x509FindType="FindBySerialNumber")
   2. Add security to binding
   3. Use added binding in endpoint (Add behaviorConfiguration="…")
   4. Specify identity for endpoint, in our case this is certificate:  
      <identity>

<certificate encodedValue="" />

</identity>

1. Go to service. Enable certificate security [docs.microsoft.com](https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/message-security-with-a-certificate-client) (Server -> Configuration)
   1. Add certificate configuration to begaviors (For now we can use same certificate, however in read environment server and client should have their own certificate)
   2. Add security to binding
2. Launch and debug. There should be exception saying that certificate cannot be verified, this is ok since certificate is “homegrown”. Real certificates cost money. To tell WCF to ignore this do the following ([stackoverflow](https://stackoverflow.com/a/4256583)):
   1. Go to client test app and add following to config (clientCredentials section):  
      <serviceCertificate>  
       <authentication certificateValidationMode="None"/>  
      </serviceCertificate>
   2. Go to service and add following to config (serviceCredentials section):  
      <clientCertificate>

<authentication certificateValidationMode="None" />

</clientCertificate>

1. Launch and debug. There should be no errors.
2. Update service configuration to trust only specific certificates (not all of them, as it is now).
   1. Add reference to IronMacbeth.Common
   2. Add trustedCertificates section to app.config

<configSections>

<section name="trustedCertificates" type="IronMacbeth.Common.CertificateValidation.TrustedCertificatesConfigurationSection, IronMacbeth.Common"/>

</configSections>

<trustedCertificates>

<add value="77F7F5D4AA2D038C4A1324DF82F1BBE4" />

</trustedCertificates>

* 1. 77F7F5D4AA2D038C4A1324DF82F1BBE4 is Serial Number of certificate currently in use. More <add value="…" /> could be added later if needed.
  2. Change client certificate validation mode to “Custom”, add customCertificateValidatorType attribute with value of "IronMacbeth.Common.CertificateValidation.IronMacbethX509CertificateValidator, IronMacbeth.Common"

1. Launch and debug. There should be no errors.