Deploying a Contract

```

Copy usingSecretNET; usingSecretNET.Common; usingSecretNET.Common.Storage; usingSecretNET.Tx;

// Select a storage provider for the wallet // Docs: https://github.com/0xxCodemonkey/SecretNET#creating--initializing-the-wallet varstorageProvider=newAesEncryptedFileStorage("","SuperSecurePassword"); varcreateWalletOptions=newCreateWalletOptions(storageProvider);

// Import wallet from mnemonic phrase // Use key created snippet "Create a new Wallet" Walletwallet=null; if(awaitstorageProvider.HasPrivateKey()) { varstoredMnemonic=awaitstorageProvider.GetFirstMnemonic(); Console.WriteLine("Use stored mnemonic: "+storedMnemonic); wallet=awaitWallet.Create(storedMnemonic,options:createWalletOptions); Console.WriteLine("wallet.Address: "+wallet.Address); }

// get infos from https://docs.scrt.network/secret-network-documentation/development/connecting-to-the-network vargprcUrl="https://grpc.testnet.secretsaturn.net"; varchainId="pulsar-3";

// Create a connection to Secret Network node // Pass in a wallet that can sign transactions // Docs: https://github.com/0xxCodemonkey/SecretNET#creating--initializing-the-wallet varsecretClient=newSecretNetworkClient(gprcUrl,chainId,wallet:wallet);

// Upload the wasm of a simple contract byte[] wasmByteCode=File.ReadAllBytes(@"mysimplecounter.wasm.gz");

// MsgStoreCode varmsgStoreCodeCounter=newMsgStoreCode( wasmByteCode, source:"",// Source is a valid absolute HTTPS URI to the contract's source code, optional builder:""// Builder is a valid docker image name with tag, optional );

varstoreCodeResponse=awaitsecretClient.Tx.Compute.StoreCode(msgStoreCodeCounter,newTxOptions() { GasLimit=2_000_000});

varcodeId=storeCodeResponse.Response.CodeId;

// contract hash, useful for contract composition varcontractCodeHash=awaitsecretClient.Query.Compute.GetCodeHashByCodeId(codeId);

// Create an instance of the Counter contract, providing a starting count varmsgInitContract=newMsgInstantiateContract( codeId:codeId, label:"My Counter {codeId}", initMsg:new{ count=100}, codeHash:contractCodeHash);// optional but way faster

varinitContractResponse=awaitsecretClient.Tx.Compute.InstantiateContract(msgInitContract,newTxOptions() { GasLimit=200_000});

//Find the contract_address in the logs varcontractAddress=initContractResponse?.Response?.Address;

varmsgExecuteContract=newMsgExecuteContract( contractAddress:contractAddress, msg:new{ increment=new{ } }, codeHash:contractCodeHash);

vartx=awaitsecretClient.Tx.Compute.ExecuteContract(msgExecuteContract,newTxOptions() { GasLimit=200_000});

```