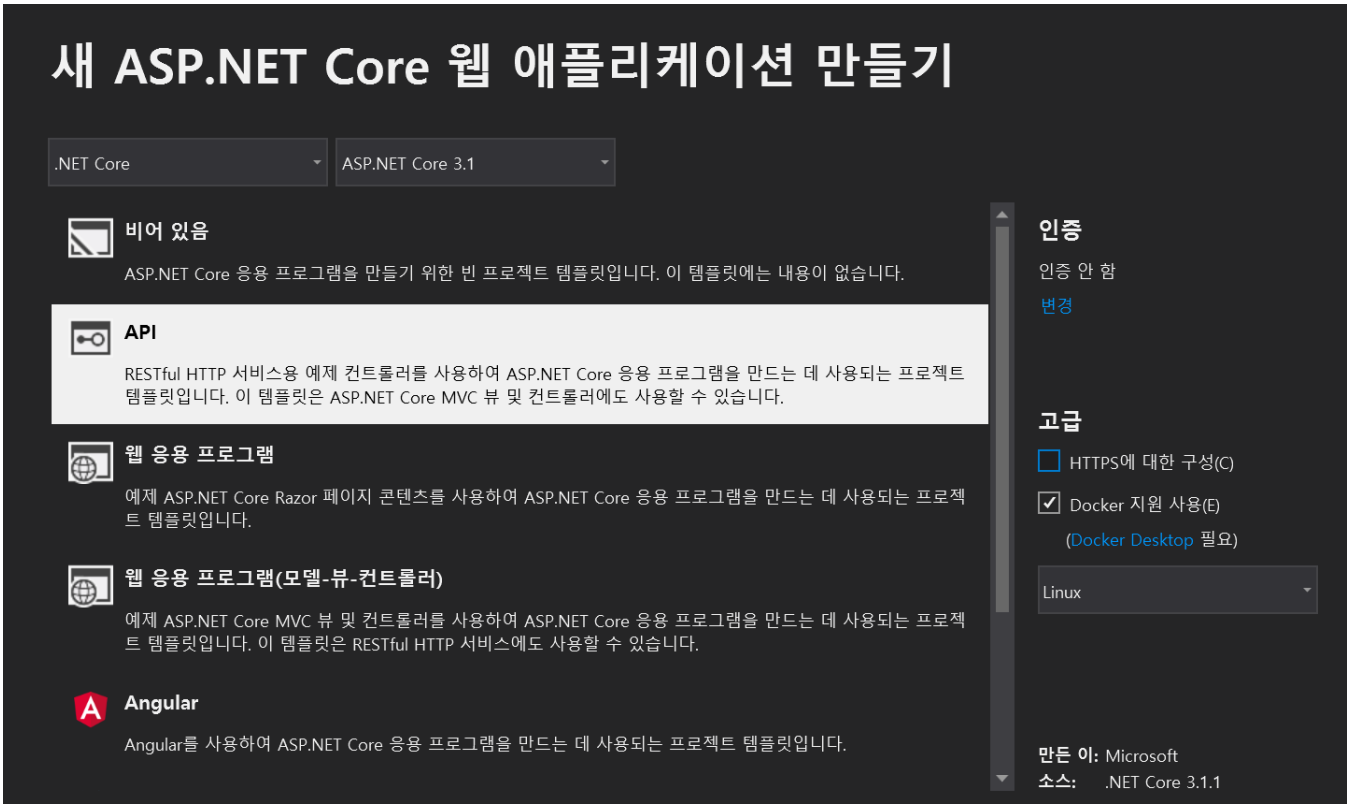
AKKA Setting For NetCore

.NET Core 3.1 API Akka.net

.

Core DI , Akka Top-Level Architecture .



.Net Core 3.1 API , Docker . ( )

AkkaNetCore .

```
<PackageReference Include="Akka" Version="1.3.17" />
<PackageReference Include="Akka.Cluster" Version="1.3.17" />
<PackageReference Include="Akka.Cluster.Tools" Version="1.3.17" />
<PackageReference Include="Akka.DI.Extensions.DependencyInjection" Version="1.3.2" />
<PackageReference Include="Akka.Logger.NLog" Version="1.3.5" />
<PackageReference Include="NLog.Web.AspNetCore" Version="4.8.1" />
```

- Akka : .
- Akka.Cluster : , .
- Akka.Cluster.Tools : . (ex>)
- Akka.DI.Extensions.DependencyInjection : .()
- Akka.Logger.NLog : Nlog .
- Nlog.Web.AspNetCore : Net Core Nlog( ) .

. NLog(https://nlog-project.org/) ,

Akka Extensions

Akka System (,) .

 , .

AkkaBoostrap -

```
using System;
using Akka.Actor;
using Akka.Cluster.Tools.Singleton;
using Microsoft.Extensions.DependencyInjection;

namespace AkkaNetCore.Extensions
{
    public static class AkkaBoostrap
    {
        public static IServiceCollection AddAkka(this IServiceCollection services, ActorSystem actorSystem)
        {
            // Register ActorSystem
            services.AddSingleton<ActorSystem>((provider) => actorSystem );
            return services;
        }

        public static IActorRef BootstrapSingleton<T>(this ActorSystem system, string name, string role = null)
where T : ActorBase
        {
            var props = ClusterSingletonManager.Props(
                singletonProps: Props.Create<T>(),
                settings: new ClusterSingletonManagerSettings(name, role, TimeSpan.FromSeconds(10), TimeSpan.
FromSeconds(3)));

            return system.ActorOf(props, typeof(T).Name);
        }

        public static IActorRef BootstrapSingletonProxy(this ActorSystem system, string name, string role,
string path, string proxyname)
        {
            var props = ClusterSingletonProxy.Props(
                singletonManagerPath: path,
                settings: new ClusterSingletonProxySettings(name, role, TimeSpan.FromSeconds(1), 100));

            return system.ActorOf(props, proxyname);
        }
    }
}
```

AkkaLoad -

```
using System;
using System.Collections.Concurrent;
using System.IO;
using System.Text;
using Akka.Actor;
using Akka.Configuration;
using Microsoft.Extensions.Configuration;
using AkkaConfig = Akka.Configuration.Config;

namespace AkkaNetCore.Config
{
    public class AkkaLoad
    {
        public static ConcurrentDictionary<string, IActorRef> ActorList = new ConcurrentDictionary<string,
IActorRef>();

        public static void RegisterActor(string name, IActorRef actorRef)
        {
            if (ActorList.ContainsKey(name)) throw new Exception("  .");
            ActorList[name] = actorRef;
        }

        public static IActorRef ActorSelect(string name)
        {
            return ActorList[name];
        }

        public static AkkaConfig Load(string environment, IConfiguration configuration)
        {
            if(environment.ToLower()!= "production")
            {
                environment = "Development";
            }

            return LoadConfig(environment, "akka{0}.conf", configuration);
        }

        private static AkkaConfig LoadConfig(string environment, string configFile, IConfiguration
configuration)
        {
            string akkaip = configuration.GetSection("akkaip").Value ?? "127.0.0.1";
            string akkaport = configuration.GetSection("akkaport").Value ?? "5100";
            string akkaseed = configuration.GetSection("akkaseed").Value ?? "127.0.0.1:5100";
            string roles = configuration.GetSection("roles").Value ?? "akkanet";

            var configFilePath = string.Format(configFile, environment.ToLower() != "production" ? string.Concat
(".", environment) : "");
            if (File.Exists(configFilePath))
            {
                string config = File.ReadAllText(configFilePath, Encoding.UTF8)
                    .Replace("$akkaport", akkaport)
                                .Replace("$akkaip", akkaip)
                                .Replace("$akkaseed", akkaseed)
                                .Replace("$roles", roles);

                var akkaConfig = ConfigurationFactory.ParseString(config);

                Console.WriteLine($"=== AkkaConfig:{configFilePath}\r\n{akkaConfig}\r\n===");
                return akkaConfig;
            }
            return Akka.Configuration.Config.Empty;
        }
    }
}
```

Config File

akka.conf / akka.Development.conf -

```
akka {
        loggers = ["Akka.Logger.NLog.NLogLogger, Akka.Logger.NLog"]
        loglevel = debug
}
```

Akka Json ,

: https://doc.akka.io/docs/akka/2.5/general/configuration.html

BasicActor - string

```
using Akka.Actor;
using Akka.Event;

namespace AkkaNetCore.Actors.Study
{
    public class BasicActor : ReceiveActor
    {
        private readonly ILoggingAdapter logger = Context.GetLogger();

        public BasicActor()
        {
            ReceiveAsync<string>(async msg =>
            {
                logger.Info($"{msg}  .");
            });
        }
    }
}
```

AkkaSystem .

```
###    : ConfigureServices
....
// *** Akka Service Setting
var envName = Environment.GetEnvironmentVariable("ASPNETCORE_ENVIRONMENT");
var akkaConfig = AkkaLoad.Load(envName, Configuration);
var actorSystem = ActorSystem.Create(SystemNameForCluster, akkaConfig);
var provider = services.BuildServiceProvider();
actorSystem.UseServiceProvider(provider);
services.AddAkka(actorSystem);


###      : Configure
app.ApplicationServices.GetService<ILogger>();
var actorSystem = app.ApplicationServices.GetService<ActorSystem>(); // start Akka.NET
ActorSystem = actorSystem;

//
actorSystem.ActorOf(Props.Create<BasicActor>(),"basic2"

//
AkkaLoad.RegisterActor(
        "basic",
        actorSystem.ActorOf(Props.Create<BasicActor>(),
        "basic"
));
```

.

> *AkkaLoad , (IActorRef) Dictionary .*
>
> *DI (https://getakka.net/articles/actors/dependency-injection.html)*
>
> *Top-level Architecture(https://getakka.net/articles/actors/dependency-injection.html)*
>
> *DI .*
>
> *(ex> x .*
>
> *. .*
>
> *, .)*

API

```
[Route("api/[controller]")]
[ApiController]
public class ActorTestController : Controller
{
        private readonly IActorRef basicActor;

        public ActorTestController()
        {
                basicActor = AkkaLoad.ActorSelect("basic");
        }

        [HttpPost("/Single/Basic/tell")]
        public void Printer(string message)
        {
                // basicActor .
                basicActor.Tell(message);
        }
...

##
[2020-03-06 22:12:43.7874] [Info] [PSMON-ASUS] [AKKA] [ACTOR] [AkkaNetCore.Actors.Study.BasicActor] [11] :   .
```

APIActor , AkkaLoad DI .

Akka.net .